TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-25-03

Acoustic-Based MAV Propeller Inspection

Zahra Khadivi · Batbayar Battseren · Wolfram Hardt

Mai 2025

Chemnitzer Informatik-Berichte

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Acoustic-Based MAV Propeller Inspection

Master Thesis

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Engineering

Submitted by: Zahra Khadivi
Student ID: 563822
Date: 31.03.2025

Supervising tutor: Prof. Dr. Dr. h. c. Wolfram Hardt
Supervising tutor: Dr. Batbayar Battseren

Abstract

The growing adoption of MAVs, alongside mobile vehicles and robots, has advanced research in autonomous driving, particularly for applications such as search and rescue operations and surveillance. However, MAV component failures can lead to mission-critical consequences. To enable fully autonomous operations, hangars can be strategically deployed to ensure MAV readiness and reliability without human intervention. A critical requirement for successful missions is verifying that MAV components, especially propellers, are undamaged and flight-ready, posing a challenge for real-time, automated inspection systems.

This thesis argues that acoustic-based fault detection offers a non-invasive, cost-effective solution for real-time propeller health monitoring in Micro Aerial Vehicles (MAVs), addressing the limitations of traditional inspection methods. Focusing on autonomous hangar deployments for emergency response scenarios like search and rescue, the study develops a methodology that integrates statistical, Mel-Frequency Cepstral Coefficients (MFCC), and Short-Time Fourier Transform (STFT) features with a hybrid ensemble of classical machine learning models, optimized using the Tree-based Pipeline Optimization Tool (TPOT). The system achieves an accuracy and F1-score of 0.9965, surpassing baseline models, and demonstrates scalability across diverse UAV models and operational conditions. By eliminating labor-intensive manual checks and resource-heavy image-based methods, the approach enhances UAV safety and operational efficiency, shifting maintenance from a reactive to a predictive paradigm. Despite challenges such as the computational cost of optimization and reliance on controlled datasets, the framework paves the way for efficient, autonomous inspection systems, with potential for transfer learning to further generalize its applicability in real-world settings.

Keywords: MAVs, Propeller, Acoustic-based Fault Detection, ML, Ensemble-learning, TPOT, Real-time Monitoring, Autonomous Hangar, Statistical Feature, MFCC, STFT.

Acknowledgments

I sincerely thank Prof. Dr. Dr. h. c. Wolfram Hardt for providing me the chance to undertake an internal master's thesis in the Department of Computer Science at Chemnitz University of Technology. His supervision and guidance were instrumental in the successful completion of this research.

I sincerely thank Dr. Batbayar Battseren for his dedicated mentorship throughout all stages of this thesis, from conceptualization to result evaluation. His invaluable feedback and continuous encouragement significantly shaped this work, fostering my growth as a researcher and enabling me to explore new concepts with confidence.

I extend special thanks to Dr. Shadi Saleh for his technical guidance in the laboratory, which greatly supported the practical aspects of this study. I am also grateful to M.Sc. Julkar Nine for her insightful feedback during the initial phase of the thesis and for inspiring the research direction, which laid a strong foundation for this work.

Additionally, I appreciate the support and inspiration provided by Dr. Shahram Khadivi, which further motivated my efforts. Finally, I am deeply thankful to my family and friends for their unwavering support and encouragement throughout this journey, making this endeavor possible.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC Analog-to-Digital Conversion

AE Amplitude Envelope

ANN Artificial Neural Network

AUC Area Under the Curve

AutoML Automated Machine Learning

BMDV German Federal Ministry for Digital and Transport

CNN Convolutional Neural Network

CRNN Convolutional Recurrent Neural Network

CSV Comma-Separated Values

CV Coefficient of Variation

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DL Deep Learning

DLRG German Lifesaving Society

DNN Deep Neural Network

DT Decision Tree

EDA Exploratory Data Analysis

F1 F1-Score

FAIR Findable, Accessible, Interoperable, Reusable

FD Fault Diagnosis

FFT Fast Fourier Transform

FN False Negatives

FP False Positives

FPR False Positive Rate

FT Fourier Transform

HNR Harmonic-to-Noise Ratio

IFC Indoor Flight Center

IMU Inertial Measurement Unit

KNN K-Nearest Neighbors

LSTM Long Short-Term Memory

MAV Micro Aerial Vehicle

MFCC Mel-Frequency Cepstral Coefficients

ML Machine Learning

MRO Maintenance, Repair, and Overhaul

MSE Mean Squared Error

PCA Principal Component Analysis

RMS Root-Mean-Square

ROC Receiver Operating Characteristic

RPM Revolutions Per Minute

RTLS Real-Time Location System

SD Standard Deviation

SRO Search and Rescue Operation

SHAP SHapley Additive exPlanations

SMOTE Synthetic Minority Oversampling Technique

SNR Signal-to-Noise Ratio

SP Signal Processing

STFT Short-Time Fourier Transform

SVM Support Vector Machine

TN True Negatives

TP True Positives

TPOT Tree-based Pipeline Optimization Tool

TPR True Positive Rate

t-SNE t-Distributed Stochastic Neighbor Embedding

UAS Unmanned Aerial System

UAV Unmanned Aerial Vehicle

USB Universal Serial Bus

WHO World Health Organization

ZCR Zero Crossing Rate

1. Introduction

Annually, drowning incidents result in the tragic loss of millions of lives and cause severe injuries to countless individuals. According to the World Health Organization (WHO), nearly 236,000 individuals worldwide succumb to drowning each year, making it a critical public health issue that demands urgent intervention [60]. In particular, inland waters, such as rivers and lakes, pose significant dangers. Recent statistics from the German Lifesaving Society (DLRG) reveal that approximately 80% of drowning fatalities occur in these inland bodies of water. In 2023, 378 drowning incidents were documented, with 90% occurring in unguarded locations, such as rivers and canals, thereby emphasizing the urgent necessity for quick rescue responses in remote areas [20]. The statistics clearly indicate the gravity of the situation and highlight the importance of rapid and efficient rescue operations, given the limited timeframe between the initial stages of drowning and the life-threatening consequences.

A robust search and rescue system, along with an autonomous deployment model, is necessary to enhance rescue capabilities and assist emergency response teams in quickly locating and aiding drowning victims in remote areas. Given the critical need for rapid response in drowning incidents, multi-copter and unmanned aerial systems are one of the areas of interest in Search and Rescue Operations (SROs) [75] [10]. They offer several advantages, particularly their ability to quickly access and survey remote or hard-to-reach areas. This ability facilitates the early identification and localization of drowning victims [71]. Moreover, UAVs can be equipped with sensors and communication systems that provide real-time data to rescue teams, enabling more informed decision-making and better coordination of rescue efforts [97].

The RescueFly Project, initiated in Germany (January 2022), introduces novel autonomous drone technology. This effort enhances water rescue operations using autonomous drone technology [69]. Funded by the German Federal Ministry for Digital and Transport (BMDV). This project seeks to shorten response times and enhance the effectiveness of rescue operations in inland waters by incorporating smart drone hangar systems that autonomously launch UAVs upon detecting distress signals. These vehicles are designed to locate drowning victims and assist them with flotation devices, thereby enhancing the overall rescue capability [34] [39] [10] [35] [87]. Altogether, these systems help to prompt response capabilities with intelligent maintenance systems, highlight the potential of UAVs to save lives

and improve public safety.

Another significant factor is that ensuring the airworthiness of UAVs is critical for their reliable performance in search and rescue operations because it directly impacts their ability to operate safely and effectively in challenging environments [52] [19]. Consequently, the reliability of UAVs heavily depends on the health and functioning of their components, including propulsion systems, navigation modules, power supplies, and communication systems [36] [29].

The propulsion system serves as a fundamental component of any aerial vehicle. Propellers are key components of these systems, play a pivotal role that ensure stable flight, maneuverability, and thrust generation. Propellers convert rotational energy into aerodynamic forces, enabling UAVs to perform essential tasks such as hovering, navigation, and payload delivery. However, they are highly susceptible to damage from collisions, regular handling, and environmental conditions, potentially resulting in reduced thrust, heightened vibrations, and impaired flight stability. Such damage can lead to reduced thrust efficiency, increased vibrations, and compromised flight stability, ultimately degrading overall system performance and reliability [80] [66] [61]. Aircraft visual inspection is designed to identify structural issues and is an integral part of Maintenance, Repair, and Overhaul (MRO) activities. It involves inspectors conducting multiple observations to ensure aircraft safety and readiness for flight [92]. Traditional propeller inspection methods rely on visual examination and manual testing, which are labor-intensive, operator-dependent, lack quantitative metrics, and are prone to subjective interpretation [92]. Besides this, image-based computer vision methods got inspired by these visual inspection to detect propellers malfunctions [36]. These methods limited to detect propeller's surface defects and may fail to detect anomalies with less lighting conditions.

Fault diagnosis processes are a key area of interest in industrial systems for ensuring system reliability. Traditional fault diagnosis methods, such as model-based approaches that compare system behavior with mathematical models and expert systems that rely on predefined rules and human expertise, are effective but often require significant computational resources and depend on accurate system models [53] [2]. In recent years, however, a significant shift toward data-driven techniques has emerged due to their ability to handle complex, nonlinear systems without requiring detailed mathematical models, addressing the critical role of accurate sensor data.

In the context of UAV fault diagnosis, researchers have explored various methodologies, ranging from vibration analysis to current signature analysis. Studies emphasize that Inertial Measurement Unit (IMU) sensor data, comprising accelerometer, gyroscope, and magnetometer measurements, are fundamental for UAV control systems, enabling the detection, isolation, and compensation of sensor faults

such as biases, noise, and outright failures [99] and [23]. However, IMUs are vulnerable to malfunctions caused by environmental influences, with accelerometer and gyroscope readings often suffering from bias and excessive noise due to temperature fluctuations and vibrations [99]. Additionally, the attitude derived from integrating gyro data as angular velocities faces significant challenges, including bias and random-walk errors [23]. Due to hardware limitations, the most common solutions, such as hardware redundancy-based methods, often present drawbacks, particularly in small UAVs, where they can compromise payload capacity and power efficiency.

In paper [13] and [61] key area of focus is the detection of faults using current sensors, which offer a practical and cost-effective means of monitoring UAV components. Current-based methods are advantageous due to the low cost and easy implementation of current sensors, as well as the integration of current and voltage detection modules in many electronic speed controllers [68]. One of the primary limitations is that current signals may not strongly reflect mechanical faults [13]. Mechanical issues such as bearing failures, rotor imbalances, or propeller damage might not always produce significant, easily detectable changes in the current signal. This is because current signals primarily capture electrical activity within the motor. Specifically, the changes in current due to mechanical faults may be subtle and difficult to distinguish from normal operational variations. This makes it challenging to diagnose mechanical faults accurately using only current-based analysis [13]. Furthermore, Fault Detection (FD) based on current can be affected by changes in operating conditions, including load, speed, and temperature. Changes in these parameters can alter the current signal, potentially masking or mimicking fault signatures, leading to false positives or negatives [68]. Additionally, they may not directly capture aerodynamic issues, such as changes in propeller efficiency due to deformation or wear, requiring more comprehensive modeling to translate electrical signals to mechanical conditions [61].

Despite their effectiveness, these methods encounter significant challenges, particularly the scarcity of reliable fault data. As [83] noted, UAV fault diagnosis poses a typical small sample problem, primarily due to the stability of UAV operations under normal conditions. This scarcity is further compounded by the risks and costs associated with inducing faults during flight tests, necessitating the exploration of hybrid methods and simulated data generation. However, simulated data often fails to replicate real-world scenarios, leaving a significant gap in fault data representation. The integration of additional sensors for FD can lead to increased hardware complexity and power consumption, impacting the UAV's operational efficiency. Consequently, there is an increasing demand for non-invasive, cost-effective, and resource-efficient approaches that can be implemented outside of UAVs.

Acoustic-based methods for inspecting UAV propellers have gained significant attention in recent years due to their non-invasive nature, cost-effectiveness, and ability to provide real-time diagnostics by leveraging the unique acoustic signatures generated under various operational conditions [80] [29]. These methods leverage the unique acoustic signatures generated by UAV propellers under various operational conditions, enabling the detection of damage, imbalance, or wear. Recent advancements in Machine Learning (ML) and acoustic analysis have facilitated the development of sophisticated techniques for diagnosing propeller faults. Acoustic sensors present a promising alternative, offering a non-invasive approach to propeller inspection. They require minimal computational resources and can provide valuable insights into propeller health without altering the UAV's payload capacity or power efficiency. Despite their advantages, acoustic-based methods must overcome challenges related to signal processing and environmental noise interference. Academic researchers often employ advanced ML and deep learning models to address these challenges, yet these models demand large datasets and significant computational power, introducing new complexities. This research aims to address some of these challenges by proposing an acoustic-based inspection method for small UAV propellers, leveraging the advantages of acoustic sensors while mitigating the limitations of traditional sensor-based approaches. The following sections will delve into the details of these systems and highlighting their potential to enhance propeller inspection and fault diagnosis in a practical and efficient manner.

Unmanned Aerial Systems

The development of Unmanned Aerial Systems (UAS) and Unmanned Aerial Vehicles (UAVs) traces its origins to the early 20th century, driven by military demands during World War I. Initially deployed as target drones, UAVs marked their first significant milestone with the Kettering Bug in 1916, widely regarded as one of the earliest UAVs [43]. Advancements during World War II introduced the Radioplane OQ-2, the first mass-produced UAV designed for target practice, further solidifying their military utility [98]. Following the war, UAV technology transitioned into civilian applications, notably in agriculture, research, and surveillance. By the 1990s, the integration of GPS technology revolutionized UAV capabilities, enabling precise navigation and expanding their use in disaster response, environmental observation, and military surveillance [98]. Within the industry, researchers have proposed various classifications to categorize UAVs, with Zakora and Molodchik offering a detailed framework based on weight and flight range, as shown in Table 1.1 [37]. This classification provides a granular perspective on the diverse UAV types employed across industries, facilitating a deeper understanding of their

applications.

Designation	**Weight Range (kg)**	**Flight Range (km)**
Micro and mini UAVs close range	$W \leq 5$	$25 \leq R \leq 40$
Lightweight UAVs small range	$5 < W \leq 50$	$10 \leq R \leq 70$
Lightweight UAVs medium range	$50 < W \leq 100$	$70 \leq R \leq 250$
Average UAVs	$100 < W \leq 300$	$150 \leq R \leq 1000$
Medium heavy UAVs	$300 < W \leq 500$	$70 \leq R \leq 300$
Heavy medium range UAVs	$500 \leq W$	$70 \leq R \leq 300$
Heavy UAVs large endurance	$1500 \leq W$	$R \leq 1500$
Unmanned combat aircraft	$500 < W$	$R \leq 1500$

Table 1.1.: Summary of drone categorization by weight and flight range [37]

In this research, we focus on Micro Aerial Vehicles (MAVs), a specialized subset of UAVs. MAVs are particularly valuable for applications demanding high agility and precision in constrained or hard-to-reach environments. While our study primarily addresses MAVs, we employ the general term "UAVs" throughout this work to maintain consistency with broader literature and industry standards. UAVs are available in multiple designs, each tailored to particular tasks. The following figure 1.1 shows Common configurations of these autonomous aerial vehicles [67].

This research focuses on Micro Aerial Vehicles (MAVs), a specialized subset of UAVs renowned for their high agility and precision in constrained or hard-to-reach environments, making them ideal for applications like search and rescue. While the study primarily addresses MAVs, the general term "UAVs" is used throughout this work to align with broader literature and industry standards. UAVs are available in multiple designs, each tailored to specific tasks, as illustrated in Figure 1.1, which presents common configurations of these autonomous aerial vehicles [67].

UAVs are equipped with a range of critical components that enable autonomous or remote-controlled operation, ensuring versatility across diverse applications. Figure 1.2 illustrates the key elements of a quadcopter, highlighting the integration of vital devices and sensors for automated vibration analysis [65]. UAV systems comprise four fundamental subsystems: airframe, propulsion, navigation, and payload mechanisms. The propulsion system, driven by propellers, generates essential aerodynamic forces through precise blade rotation, creating differential pressure zones that enable controlled flight operations [28]. Modern UAV architectures further enhance operational capabilities by incorporating sophisticated sensor arrays, communication interfaces, and autonomous control systems [74].

The propulsion system of UAVs, particularly in multi-rotor configurations, plays a critical role in determining flight stability, maneuverability, and payload capacity.

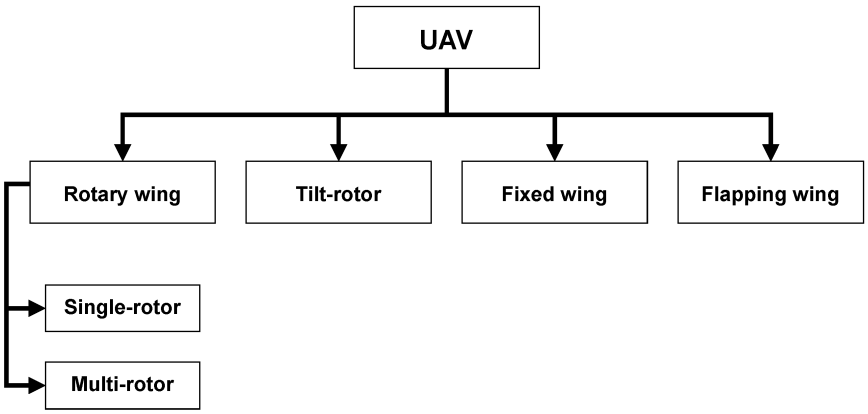


Figure 1.1.: Different UAVs categories [67]

Multi-rotor UAVs are characterized by diverse propeller arrangements, which vary based on the number of rotors and their spatial distribution, as illustrated in Figure 1.3.

The propeller mechanism, serving as the critical propulsion element, utilizes advanced aerodynamic principles to generate thrust through rotational blade dynamics. This process creates controlled pressure differentials across blade surfaces, enabling sustained flight operations and stability control [28] [54]. In challenging operational environments, propeller performance directly correlates with system reliability and mission success. Performance degradation can manifest through various mechanical phenomena: structural damage (cracks, surface defects), material fatigue, and geometrical deformations [28] [63]. These degradation mechanisms induce system-wide effects, including vibrational anomalies, flight path instabilities, and reduced propulsive efficiency, potentially compromising mission parameters. Common types of propeller damage include cracks, chips, erosion, and deformation, often caused by collisions, environmental factors, or material fatigue. Fractures and nicks, often caused by collisions with obstacles or debris, can modify the propeller's aerodynamic characteristics, resulting in heightened vibrations and decreased thrust performance [66]. Erosion, caused by prolonged exposure to abrasive particles such as sand or dust, gradually wears down the propeller surface, diminishing its structural integrity and performance [16]. Deformation, often due to excessive mechanical stress or thermal effects, can cause imbalances and misalignment, further exacerbating operational inefficiencies [95]. These damages not only compromise the UAV's flight capabilities but also pose safety risks, emphasizing the need for regular inspection, robust material selection, and advanced damage detection techniques to mitigate such issues.

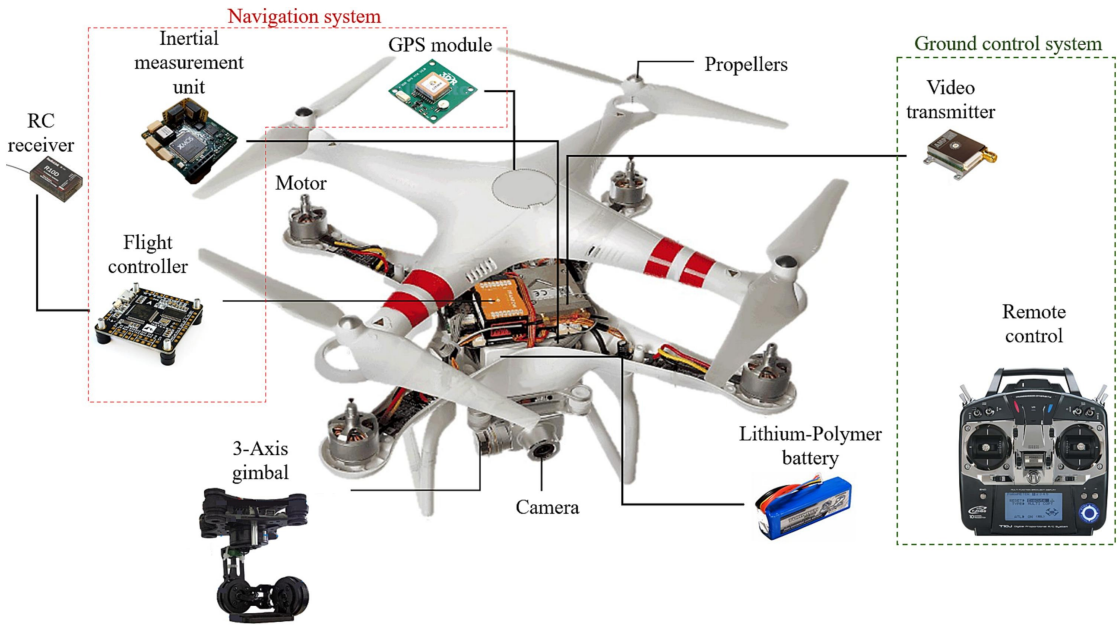


Figure 1.2.: UAV components [65]

Aerodynamic Noise

Aerodynamic noise generated by propeller-driven aircraft represents a complex acoustic phenomenon that significantly influences both environmental impact and system diagnostics, offering a critical lens for understanding propeller health. The acoustic signature of propeller systems comprises two primary characteristics: discrete frequency tonal components and broadband noise contributions, as evidenced by studies on non-destructive evaluation of UAV propellers [29]. Understanding these acoustic elements is essential for developing reliable propeller inspection methodologies, particularly in the context of acoustic-based fault detection.

Tonal components, which emerge from periodic blade motion, manifest as distinct peaks in the frequency spectrum at the blade passing frequency and its harmonics, directly correlating with rotational speed and the number of blades to provide valuable diagnostic insights into propeller condition. Comprehensive aircraft noise studies demonstrate that these tonal signatures are particularly pronounced in propeller-driven systems due to the periodic nature of blade-air interactions [24] . In contrast, broadband noise arises from complex turbulent flow interactions, especially at blade edges and tip regions, appearing as a broad range of frequencies driven by unpredictable pressure changes in the turbulent boundary layer and wake areas. These broadband characteristics are influenced by aerodynamic factors such as blade geometry, angle of attack, and local flow conditions, becoming particularly relevant in urban environments where complex flow patterns

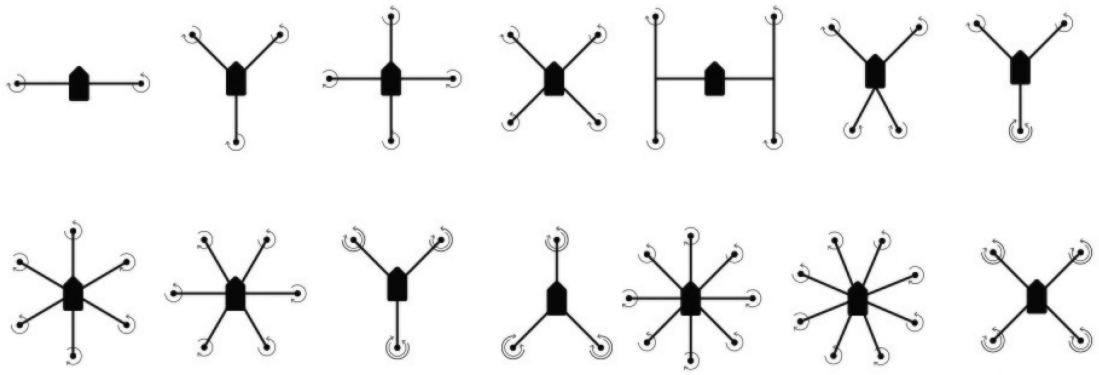


Figure 1.3.: Types of multirotor propeller arrangements, featuring (top row, left to right) bicopter, tricopter, quad +, quad X, quad H, quad V, and quad Y, and (bottom row, left to right) hexa +, hexa X, hexa Y6, hexa IY, octo +, octo X, and octo X8 [65].

can significantly alter the acoustic signature [78].

The interaction between tonal and broadband components creates a distinctive acoustic profile that varies with operational conditions and propeller health status, enabling modern acoustic analysis techniques to leverage both for diagnostic purposes. Tonal elements often reveal systematic variations in blade geometry or rotation, while broadband characteristics can indicate surface degradation or flow disturbances, forming the foundation for robust acoustic-based inspection methodologies.

1.1. Motivation

The development of an automated, lightweight, and acoustic-based FD system for UAV propellers is driven by critical challenges in UAV maintenance and operational safety. Current propeller inspection methods suffer from significant limitations, including invasiveness, high computational demands, and an inability to detect subtle damages that could compromise UAV performance, potentially leading to mission failure. This research proposes a non-destructive, cost-effective, and real-time monitoring approach to identify propeller health across diverse damage scenarios without imposing additional computational or physical burdens on UAV systems. By leveraging acoustic signal processing and machine learning techniques, the study seeks to shift propeller health monitoring from a reactive to a predictive maintenance paradigm, filling a critical gap in UAV diagnostics and preventing catastrophic failures through early damage detection. Autonomous UAV inspection systems, such as those developed for high-voltage transmission line monitor-

ing, demonstrate the potential for automated maintenance in critical infrastructure [85]. Recent advancements in adaptive UAV platforms for autonomous inspection of high-voltage power lines further highlight the growing role of UAVs in critical applications, emphasizing the need for reliable component health monitoring to ensure mission success [7].

Acoustic-based UAV propeller inspection has emerged as a promising approach for ensuring the safety and reliability of UAV operations. Through the sophisticated feature extraction methods, machine learning algorithms, and varied datasets, researchers have achieved notable advancements in identifying propeller irregularities. However, challenges such as noise interference and computational efficiency must be addressed to fully realize the potential of these methods. As the field continues to evolve, the integration of acoustic-based diagnostics into routine maintenance protocols will play a pivotal role in enhancing UAV performance and reducing operational risks. In addition to propeller health monitoring, collision avoidance represents a critical safety concern for UAVs in autonomous operations, particularly in confined spaces like hangars, as explored in studies on robust perception depth information [72].

Traditional propeller inspection methods, such as visual examination and manual testing, are labor-intensive and prone to subjective interpretation, lacking quantitative metrics and often failing to detect subtle damages that are not visually apparent [92]. Similarly, image-based computer vision methods, though inspired by visual inspection, are limited to detecting surface defects and may fail under poor lighting conditions [36]. In contrast, acoustic-based methods provide a non-invasive approach, requiring minimal computational resources while offering valuable insights into propeller health without affecting the UAV's payload capacity or power efficiency [80]. Nevertheless, these methods must address challenges related to signal processing and environmental noise interference to ensure reliable performance in real-world scenarios.

The proposed system addresses these challenges by analyzing acoustic signatures generated by UAV propellers to detect subtle faults, such as cracks, blade deformations, or aerodynamic imbalances, before they lead to system failure. By employing statistical feature extraction and machine learning models, this research develops a cost-effective, energy-efficient solution that operates in real time, enabling UAVs to function autonomously without significant battery consumption or computational overhead. The system will be rigorously tested across various operational conditions, simulating real-world scenarios where UAVs encounter dynamic and noisy environments, using audio recordings captured from propellers operating at different speeds and under diverse damage scenarios. This approach ensures the system's robustness in identifying faults across a wide range of conditions. Furthermore, the study will compare the performance of acoustic analysis

with conventional fault detection techniques, such as vibration or sensor-based systems, highlighting the advantages of acoustic monitoring, including simpler implementation, lower costs, and real-time capabilities.

This research focuses on developing and validating a fault detection model based on acoustic signatures, tested on two UAV types under varied conditions, including different propeller speeds and damage levels. Designed for real-time processing on embedded platforms, the model will also be evaluated on cloud-based systems to assess its compatibility with resource-intensive setups. Primarily intended for autonomous UAVs in emergency response operations, the model holds potential for extension to fields like environmental monitoring and agriculture, demonstrating the versatility of acoustic fault detection. The study underscores the practicality of this approach in dynamic, real-world UAV deployments, offering a viable alternative to conventional methods. However, the research is limited to prototype UAV systems and may not fully address challenges associated with large-scale implementation or highly noisy environments. Ultimately, the findings aim to advance UAV health monitoring, paving the way for more efficient and dependable autonomous systems in critical industries.

1.2. Objective and Scope

This study aims to explore how recording procedures and experimental setups shape the development of an effective data-driven FD model for UAV propellers, focusing on identifying key acoustic features and establishing implementation guidelines for drone hangar environments. The primary objective is to design a lightweight, real-time FD solution deployable on embedded UAV platforms, ensuring operational reliability for critical missions such as search and rescue. By analyzing acoustic signatures, the system seeks to detect minor faults—such as cracks, blade deformations, and aerodynamic imbalances—before they escalate into system failures, utilizing statistical feature extraction and ML techniques to create a cost-effective, energy-efficient solution that enables UAVs to operate autonomously with minimal computational or power demands [80]. The system will be rigorously tested under diverse operational conditions, simulating real-world scenarios where UAVs encounter dynamic and noisy environments, with data collected from audio recordings of propellers at varying speeds and damage levels to ensure robust fault identification across a range of states. Furthermore, the research will compare the performance of acoustic-based FD with conventional methods, such as vibration analysis or sensor-based systems, highlighting the advantages of acoustic monitoring in terms of simplicity, cost-efficiency, and real-time capabilities [92].

This research focuses on developing and validating an acoustic-based FD model, with testing conducted on two UAV types under varying propeller speeds and

damage conditions. Optimized for real-time processing on embedded platforms, the model will also be evaluated on cloud-based systems to assess its compatibility with resource-intensive setups. While primarily designed for autonomous UAVs in emergency response operations, the model offers potential applications in sectors like environmental monitoring and agriculture, highlighting the versatility of acoustic FD [80]. The study demonstrates the practicality of this approach in real-world UAV operations within dynamic environments, positioning acoustic FD as a viable alternative to traditional methods. However, the research is limited to prototype UAV systems and may not fully address challenges associated with large-scale deployments or highly noisy settings. Ultimately, the outcomes will advance UAV health monitoring, fostering the development of more efficient and dependable autonomous systems for critical applications.

1.3. Thesis Structure

This thesis is organized into several chapters, each building upon the previous one to provide a comprehensive overview of the research, its methods, and its outcomes. The structure is outlined as follows:

textbfChapter 1: Introduction

Chapter 1 introduces the challenge of propeller malfunction detection in UAVs, with a focus on emergency response scenarios. It provides background context, articulates the research's significance, and details the objectives and motivation for the proposed acoustic-based solution.

Chapter 2: Background Knowledge

Chapter 3 reviews existing research on fault detection systems for UAVs, analyzing the strengths and limitations of various methods, with a particular focus on acoustic-based approaches and machine learning techniques for propeller fault detection.

Chapter 3: State of the Art

Chapter 3 reviews existing research on fault detection systems for UAVs, analyzing the strengths and limitations of various methods, with a particular focus on the application and evaluation of acoustic signatures in state-of-the-art techniques.

Chapter 4: System Design

Chapter 4 presents the conceptual framework for the acoustic-based fault detection system, detailing the methodology overview, data acquisition, preprocessing pipeline, feature extraction, and model training processes employed to classify UAV propeller health.

Chapter 5: Methodology

Chapter 5 explores the practical implementation of the system, providing an in-depth examination of the system setup, experimental configuration, software

platforms, audio data preparation, preprocessing pipeline, data processing, model training, and optimization steps for embedding the fault detection system into UAVs.

Chapter 6: Implementation

Chapter 6 evaluates the performance of the acoustic fault detection system, presenting classification outcomes, comparing the two modeling approaches, and assessing the optimized hybrid ensemble's effectiveness, with a focus on accuracy and reliability across diverse conditions.

Chapter 7: Results and Evaluation

Chapter 7 interprets the results, discussing the effectiveness of the proposed approaches, their strengths and limitations, and their implications for real-world UAV applications. It also proposes future research directions, such as transfer learning, to enhance generalization and scalability.

Chapter 8: Conclusion and Future Work

Chapter 8 concludes the research by summarizing key findings, emphasizing the system's effectiveness for autonomous UAV inspection, and addressing limitations to guide future improvements.

2. Background Knowledge

This chapter establishes the theoretical foundation for the acoustic-based fault detection system for UAV propellers, focusing on signal processing, audio signal processing, and machine learning techniques. These principles form the basis for the analysis, preprocessing, and classification methods employed in this thesis.

2.1. Sound Waves

Sound waves are mechanical vibrations that travel through a medium, by creating localized compressions and rarefactions of the medium's particles. Unlike electromagnetic waves, sound waves necessitate a physical medium to transmit energy, rendering them dependent on molecular interactions for energy transfer from the source to the receiver. These waves encapsulate vital information about their originating vibrating source, making their comprehension indispensable for applications in acoustic analysis and audio signal processing [6] [62]. As sound waves propagate, particles within the medium oscillate around their equilibrium positions, transferring energy through intermolecular forces without causing permanent displacement. This oscillatory behavior enables sound waves to efficiently convey energy and information, forming the basis for their analysis in engineering and scientific domains. The properties of sound waves are characterized by several key attributes:

- **Wavelength (λ):** The distance between consecutive compressions or rarefactions in a wave. Wavelength is inversely related to frequency, as described by the equation:

$$\lambda = \frac{v}{f} \tag{2.1}$$

  where v is the velocity of sound and f is the frequency. Longer wavelengths correspond to lower frequencies and produce lower-pitched sounds, while shorter wavelengths correspond to higher frequencies and higher pitches.

- **Frequency (f):** The number of wave cycles passing a fixed point per second, measured in hertz (Hz). Frequency determines the pitch of a sound. The human ear typically detects frequencies between 20 Hz and 20 kHz, known as

the audible range. Frequencies beyond this range are classified as ultrasonic. The relationship between frequency and the time period (T) is expressed as:

$$f = \frac{1}{T} \tag{2.2}$$

- **Time Period (T):** The time it takes for one complete cycle of the wave to pass a given point. It is inversely related to frequency.
- **Velocity (v):** The speed at which sound travels through a medium. This velocity depends on factors such as the medium's temperature, density, and elasticity. In air at room temperature (approximately 20°C), sound travels at a velocity of approximately 343 meters per second. The velocity can also be expressed as:

$$v = \lambda \cdot f \tag{2.3}$$

- **Amplitude (A):** The maximum displacement of particles from their rest position. Amplitude is perceived as the loudness of a sound; greater amplitude corresponds to louder sounds.

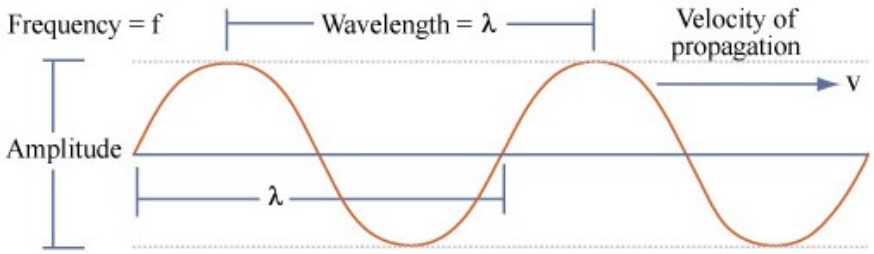


Figure 2.1.: Illustration of sound wave properties [59]

These properties collectively describe how sound waves propagate and interact with their environment, providing a foundation for acoustic-based ML systems. For instance, wavelength and frequency enable spectral feature extraction, while amplitude and velocity support intensity and localization analyses for detecting faults in UAV propellers.

2.2. Signals

A signal is a mathematical function of one or more independent variables that encodes information about a physical phenomenon. It is often represented over time

or space. An audio signal, specifically, represents sound and is typically captured as variations in air pressure over time. An audio signal is a representation of sound, typically captured as variations in air pressure over time. Audio signals encode all the information necessary to reproduce sound, making them essential for applications in speech processing, music analysis, and acoustic fault detection [6]. Signals are categorized based on the nature of their independent variables, primarily into continuous-time and discrete-time types. This classification determines the methods used for their analysis and processing. A signal is considered continuous-time if it is defined over an unbroken continuum of values, whereas discrete-time signals are defined only at specific, countable intervals. This distinction underpins many methods of signal analysis and processing, influencing how data is modeled and interpreted in diverse applications.

- **Analog Signals:** Analog signals represent continuous waveforms that vary smoothly over time and amplitude. They are inherently time-varying, with their values capable of taking on any real number within a defined range [25]. However, analog signals require infinite memory for precise representation, which is impractical for digital systems. This limitation necessitates the conversion of analog signals into digital form for processing and storage.
- **Digital Signals:** Digital signals, on the other hand, consist of discrete values, usually represented as binary sequences of 0s and 1s. These signals are generated from analog sources through a process known as Analog-to-Digital Conversion (ADC) [82]. Unlike analog signals, digital waveforms exist within a finite set of possible states during a specific time interval, making them ideal for computational analysis and storage. This discretization is crucial for enabling efficient data processing and storage.

The distinction between continuous and discrete signals is fundamental for choosing appropriate processing techniques, such as Analog-to-Digital Conversion (ADC), which plays a key role in preprocessing. This classification guides the transformation of raw audio data into a format suitable for ML applications.

2.3. Signal Processing

Signal processing (SP) involves performing operations on signals to either modify them in a desired way or extract valuable information. It combines theoretical concepts, design strategies, and practical implementation to convert signals from various sources into a form that allows for effective analysis and utilization [6]. In computational applications, continuous signals are often converted into discrete forms through sampling at uniform intervals. The ADC process involves two key

steps, sampling the analog signal at specific time intervals and quantizing the amplitudes into distinct levels [82].

Sampling

Sampling involves capturing the amplitude of an analog signal at specific time intervals, converting it into discrete points. The sampling rate (S_r) determines how often samples are taken from the analog signal and is measured in samples per second or hertz (Hz). A higher sampling rate results in more samples per second, providing a more accurate representation of the original analog signal but requiring greater memory resources. Conversely, a lower sampling rate captures fewer samples per second, leading to a less precise representation but reducing memory usage. A higher sampling rate minimizes sampling error, which is the discrepancy between the original analog signal and its digital representation [70]. The Nyquist-Shannon Frequency (f_N), defined as half the sampling rate, represents the highest frequency that can be accurately reconstructed from the sampled signal [70]:

$$f_N = \frac{S_r}{2} \tag{2.4}$$

Frequencies exceeding (f_N) can cause aliasing, where higher-frequency components are incorrectly represented as lower frequencies, introducing artifacts into the reconstructed signal [70].

Quantization

Quantization approximates the sampled amplitudes to the nearest value within a predefined finite set, enabling digital representation. This process introduces quantization error, which is the difference between the original amplitude and its quantized value. The resolution of quantization depends on the number of bits used to represent each sample, with higher bit depths reducing quantization error and improving signal fidelity. The Signal-to-Noise Ratio (SNR) measures the relationship between the maximum signal strength and the quantization error, providing a metric for the dynamic range of the digital signal [70].

Framing

Framing is a crucial preprocessing step in audio signal processing, where a continuous audio signal is divided into short, manageable segments called frames. This technique is essential for analyzing non-stationary signals, where characteristics like frequency content or amplitude vary over time. By breaking the signal into

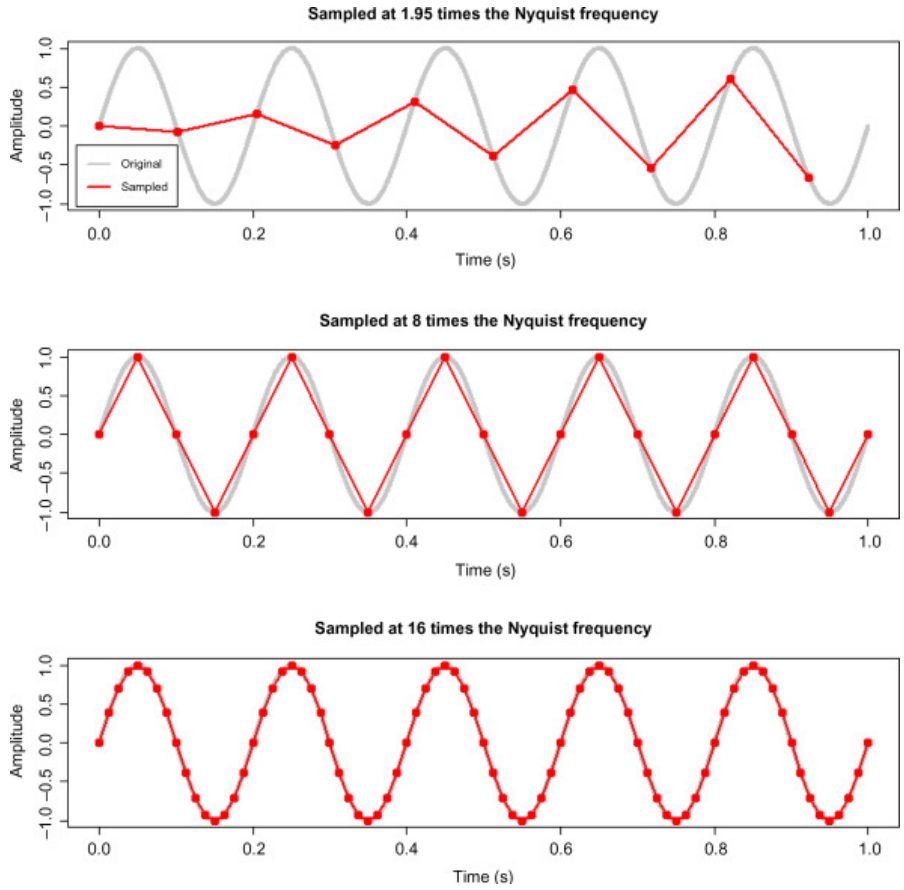


Figure 2.2.: A signal with a Nyquist frequency of 5 Hz, sampled at different rates. Sampling below the Nyquist frequency causes aliasing, making the original signal unrecoverable. A sampling rate 16 times higher reconstructs the signal with greater detail.[77]

smaller segments, framing allows for the application of analysis techniques that assume local stationarity, aligning with the temporal resolution of human auditory perception and the requirements of ML systems [47].

The need for framing arises from the dynamic nature of audio signals. Unlike stationary signals, real-world audio signals exhibit variations due to changes in the sound source or environmental conditions. Analyzing the entire signal as a single entity would obscure these time-varying features. Framing addresses this by segmenting the signal into overlapping or non-overlapping frames, enabling detailed temporal analysis. Overlapping frames are commonly used to ensure continuity and prevent information loss at frame boundaries. The degree of overlap is defined by the hop length, typically set to 25–50

After framing, each segment is often processed further, such as through win-

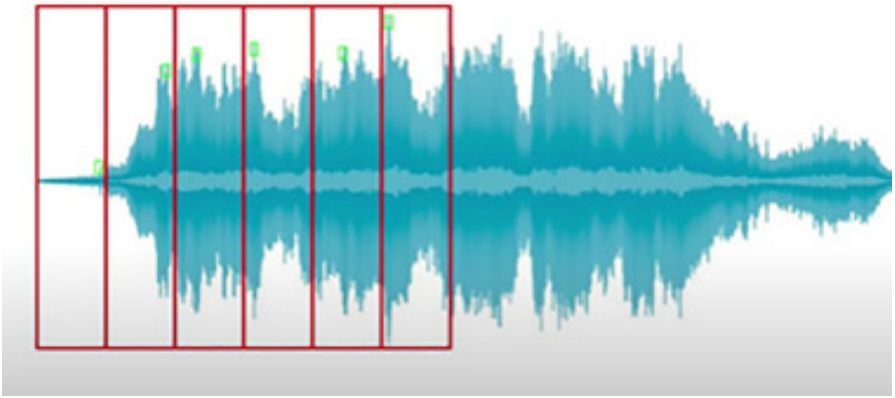


Figure 2.3.: Decomposition of audio signals into frames. [47]

dowing, to reduce edge effects before applying transformations like the Fourier Transform. Framing thus bridges raw signal acquisition and feature extraction, enabling the identification of temporal patterns, such as sudden amplitude spikes or frequency shifts, that indicate faults in UAV propellers.

Windowing and Spectral Leakage

When applying the Fourier Transform, the signal often does not consist of an integer number of periods, leading to spectral leakage. This occurs when discontinuities at the signal's endpoints introduce high-frequency components absent in the original signal. To mitigate this, windowing is applied to each frame before the Fourier Transform. Windowing functions, such as the Hann window, taper the signal at the edges, reducing discontinuities and minimizing spectral leakage. However, windowing results in the loss of information at frame boundaries. To address this, overlapping frames are used, where consecutive frames overlap by a specified number of samples (the hop length), ensuring no information is lost [48].

$$x_w[n] = x[n] \cdot w[n] \tag{2.5}$$

Here, $w[n]$ represents the window function. While windowing improves spectral clarity, it slightly reduces resolution by weighting edge samples less heavily. This trade-off is managed by adjusting frame overlap and the type of window used.

2.4. Audio Features

Audio features are descriptors that capture various characteristics of sound, enabling ML systems to identify patterns and classify audio signals. Different features

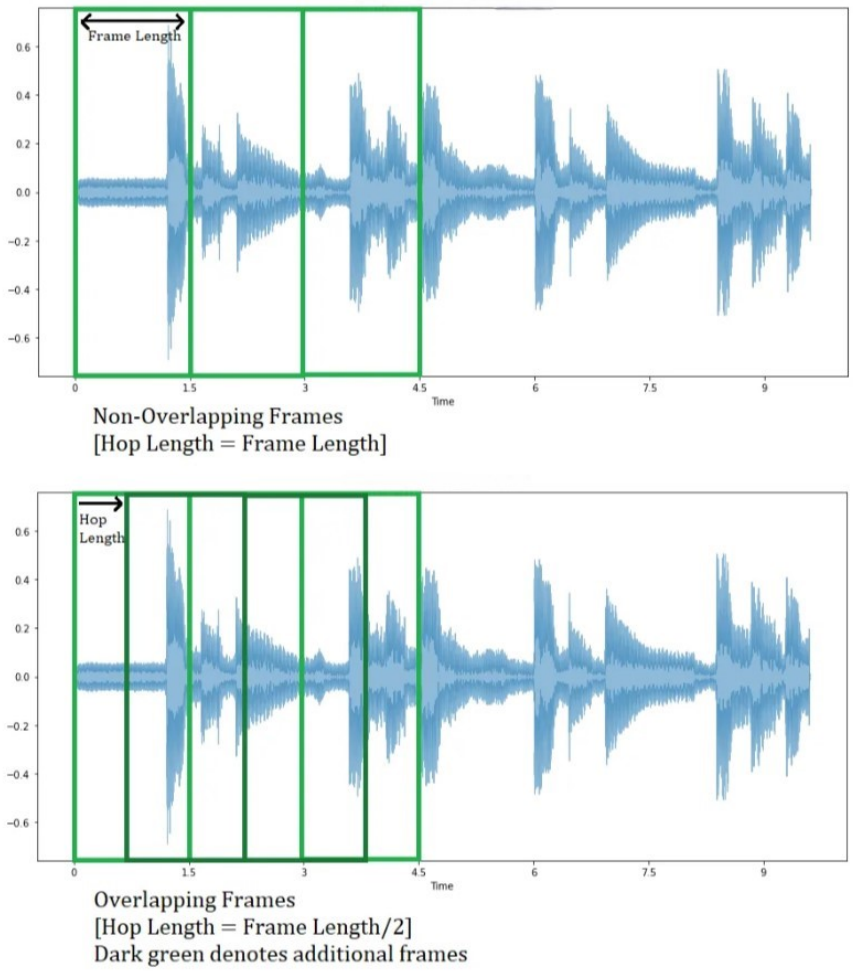


Figure 2.4.: Illustration of the overlapping effect [73]

highlight distinct aspects of sound, and they are broadly categorized into time-domain and frequency-domain features. Each category provides complementary insights: time-domain features emphasize temporal dynamics, while frequency-domain features reveal spectral composition. This dual approach enhances the robustness of acoustic analysis, ensuring a comprehensive understanding of the signal's properties.

2.4.1. Time-Domain Features

Time-domain features come straight from the audio signal's raw waveform, with time on the x-axis and amplitude on the y-axis. They capture temporal aspects like energy, zero-crossing rate, and statistical measures without needing a shift to

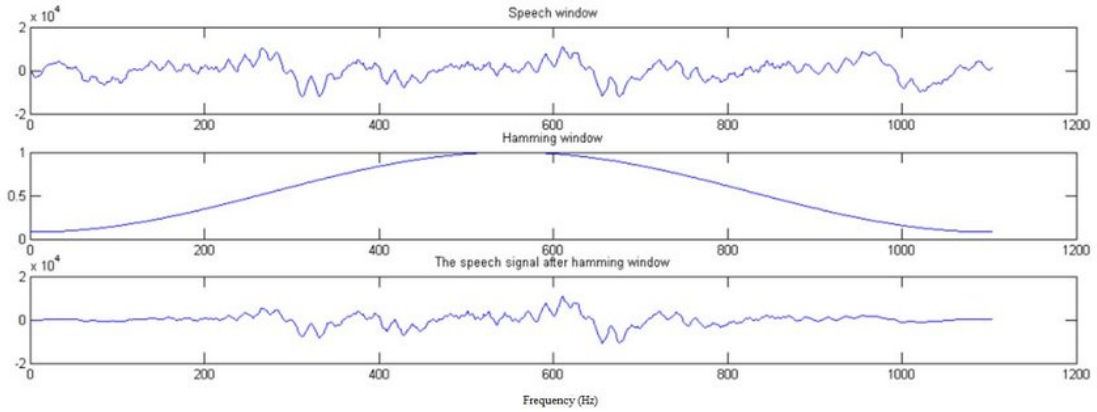


Figure 2.5.: (a) Audio frame, (b) 25 ms hamming window, (c) Marked status of the audio signal [84]

the frequency domain. This direct method is fast and effective for spotting sudden changes, such as shifts in UAV propeller sounds due to faults. To extract these features, the signal is digitized via sampling and quantization, then divided into frames short windows, typically 10–50 milliseconds long. Each frame is examined separately, allowing the system to detect time-based variations that might point to issues like cracks, imbalances, or wear in propeller blades.

Amplitude Envelope

The amplitude envelope (AE) represents the maximum amplitude value within a frame, providing a straightforward estimate of the signal's loudness. It reflects the peak intensity of the sound during that time segment. Mathematically, for a frame with samples $x[n]$, where $n = 1, 2, ..., N$, the AE is defined as:

$$\text{AE} = \max(|x[n]|) \tag{2.6}$$

where $|x[n]|$ is the absolute value of the sample amplitude. This feature is particularly useful for onset detection, identifying abrupt increases in amplitude that might correspond to mechanical impacts or irregularities in UAV propeller operation. However, AE is highly sensitive to outliers, such as noise spikes or transient distortions, which can skew its representation of loudness. This limits its reliability in noisy environments. For fault detection, AE can highlight sudden events, but its susceptibility to interference necessitates the use of complementary features for robust analysis.

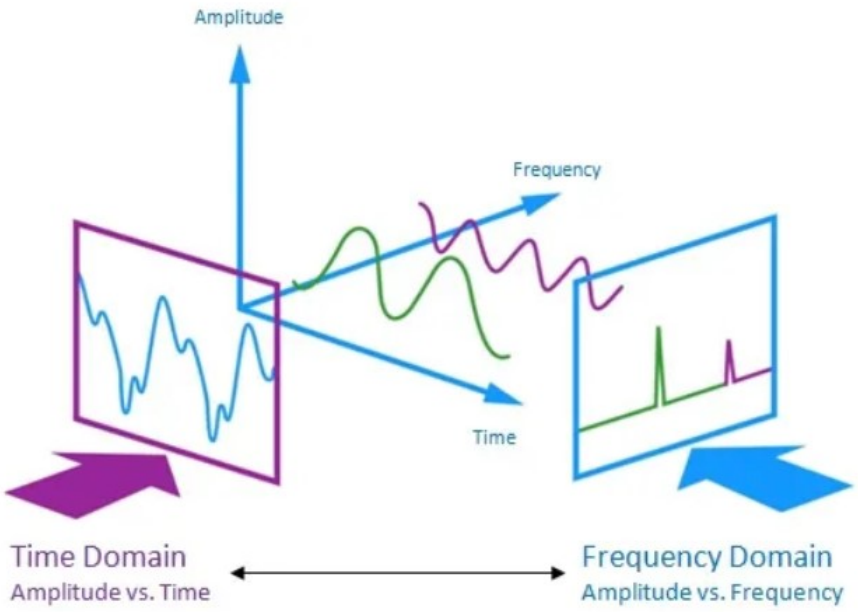


Figure 2.6.: Illustration of audio features [73]

Root-Mean-Square Energy

The Root-Mean-Square (RMS) energy measures the overall energy of a frame by computing the square root of the mean of squared amplitudes across all samples. For a frame of N samples $x[n]$, RMS is calculated as:

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x[n]^2} \tag{2.7}$$

RMS serves as a robust indicator of loudness, averaging the signal's energy over the frame to provide a more stable measure than the amplitude envelope. Unlike AE, RMS is less sensitive to outliers because it considers the contribution of all samples, making it a reliable descriptor for assessing the intensity of UAV propeller sounds under varying conditions. In audio signal processing, RMS is widely used for segmentation tasks, such as distinguishing between silent and active periods. In fault detection, it can reveal changes in energy patterns associated with propeller damage, such as increased vibration intensity due to structural faults.

Zero Crossing Rate

The Zero Crossing Rate (ZCR) quantifies the number of times the signal crosses the horizontal axis (zero amplitude) within a frame, reflecting the frequency of

sign changes in the waveform. For a frame of N samples $x[n]$, ZCR is computed as:

$$\text{ZCR} = \frac{1}{N-1} \sum_{n=1}^{N-1} |\text{sign}(x[n]) - \text{sign}(x[n-1])| \tag{2.8}$$

where | is an indicator function that returns 1 if the product $x[n] \cdot x[n+1]$ is negative (indicating a sign change) and 0 otherwise. ZCR is a versatile feature, capturing the signal's oscillatory behavior. It excels at distinguishing percussive sounds (high ZCR, rapid transitions) from pitched sounds (lower ZCR, smoother oscillations), making it valuable for monophonic pitch estimation and voiced/unvoiced decisions in speech processing. In the context of UAV propellers, ZCR can detect irregularities in rotational patterns, such as erratic vibrations from a damaged blade, that manifest as increased zero crossings compared to the steady hum of a healthy propeller. This sensitivity to temporal fluctuations enhances its utility in fault detection systems.

2.4.2. Frequency-Domain Features

Frequency-domain features focus on the spectral composition of an audio signal, emphasizing its frequency components rather than its temporal evolution. These features are obtained by applying the Fourier Transform to the time-domain waveform, converting it into the frequency domain and producing a spectrogram. The spectrogram visualizes the signal's frequency content over time: the x-axis represents time, the y-axis denotes frequency, and color intensity indicates the magnitude of each frequency component. This representation is particularly effective at revealing harmonic structures, periodic patterns, and subtle anomalies, attributes that are critical for diagnosing faults in UAV propellers. By shifting the focus from amplitude variations to spectral characteristics, frequency-domain features enable the identification of changes in the acoustic signature that may arise from structural damage, such as blade imbalances or wear, making them indispensable for advanced audio analysis.

Spectral Centroid

The spectral centroid represents the "center of mass" of the frequency spectrum within a frame, summarizing the distribution of frequency energy. It is calculated as the weighted average of frequencies, with weights determined by their magnitudes:

$$\text{Spectral Centroid} = \frac{\sum_{k=0}^{K-1} f[k] \cdot |X[k]|}{\sum_{k=0}^{K-1} |X[k]|} \tag{2.9}$$

where $f[k]$ is the frequency corresponding to bin k, $|X[k]|$ is the magnitude from the Fourier Transform, and K is the number of frequency bins. Expressed in hertz, the spectral centroid indicates the perceived brightness of a sound higher values correspond to sharper, higher pitched tones, while lower values suggest duller, bass-heavy sounds. In the context of UAV fault detection, shifts in the spectral centroid can signal alterations in propeller vibration patterns, such as those caused by mass imbalances or surface irregularities, providing a sensitive marker for anomaly detection.

Spectral Flux

Spectral flux measures the rate of change in the power spectrum between consecutive frames, quantifying how rapidly the frequency content evolves over time. It is defined as the squared difference in magnitude across frames:

$$\text{Spectral Flux} = \sum_{k=0}^{K-1} (|X_t[k]| - |X_{t-1}[k]|)^2 \tag{2.10}$$

where $X_t[k]$ and $X_{t-1}[k]$ represent the magnitudes of the current and previous frames, respectively, for frequency bin k. This feature excels at detecting transitions or disruptions in the signal, such as sudden shifts in frequency distribution caused by a propeller fault. For UAV applications, spectral flux can highlight dynamic anomalies, like the onset of a crack or a loose component, that alter the spectral stability, enhancing the system's ability to identify faults that may not be evident in static features.

Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) are a set of coefficients that encapsulate the tonal characteristics of sound, derived from the frequency domain through a process designed to emulate human auditory perception [1]. Unlike raw spectral analysis, MFCCs emphasize perceptually significant features by transforming the signal into a compact representation. The computation involves several steps: applying a Fourier Transform to obtain the power spectrum, mapping this spectrum onto the Mel scale, a logarithmic scale that prioritizes lower frequencies to reflect human hearing, taking the logarithm of the Mel filterbank outputs, and applying a Discrete Cosine Transform (DCT) to produce a concise set of coefficients. This process captures the signal's spectral envelope efficiently, making MFCCs robust to noise and effective for distinguishing complex audio patterns.

The MFCCs are mathematically defined as:

$$c_n = \sum_{k=1}^{K} \log(S_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right], \quad n = 0, 1, \ldots, N-1 \tag{2.11}$$

where:

- c_n: The n-th MFCC coefficient,
- S_k: The log-energy output of the k-th Mel filterbank,
- K: The number of filterbank channels,
- n: The coefficient index (typically 0 to 13–20),
- N: The total number of coefficients computed.

Typically, the first 13–20 coefficients are retained, with lower-order coefficients (e.g., c_0, c_1, c_2) describing the overall spectral shape and higher-order coefficients capturing finer spectral details. In the context of UAV propeller acoustics, MFCCs are particularly valuable for their ability to isolate harmonic signatures, such as those distinguishing intact propellers from those with structural defects like cracks or deformations gourisaria2024comparative. Their noise resilience and sensitivity to subtle spectral shifts make them a cornerstone for audio-based fault detection, providing a robust foundation for classifying propeller conditions.

Short-Time Fourier Transform

The Short-Time Fourier Transform (STFT) is a powerful transformation that combines time and frequency analysis by applying the Fast Fourier Transform to short, overlapping segments of a signal. Unlike the standard Fourier Transform, which provides a global frequency representation, STFT preserves temporal information by analyzing the signal through a sliding window, producing a spectrogram, a two-dimensional map of frequency content over time LACOSTE201079. This makes STFT particularly suited for non-stationary signals, such as those generated by UAV propellers, where acoustic characteristics vary with rotational speed or structural conditions.

The STFT of a continuous-time signal $x(t)$ is defined as:

$$X(\tau, f) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-j2\pi f t}\, dt \tag{2.12}$$

where:

- $X(\tau, f)$: The STFT output at time τ and frequency f,
- $x(t)$: The input signal,
- $w(t-\tau)$: A window function centered at time τ,
- $e^{-j2\pi ft}$: The complex exponential for frequency decomposition,
- $j = \sqrt{-1}$: The imaginary unit.

For digital signals, the discrete STFT is computed as:

$$X[m,k] = \sum_{n=0}^{N-1} x[n]w[n-mH]e^{-j2\pi kn/N} \tag{2.13}$$

where:

- $X[m,k]$: The STFT at time frame m and frequency bin k,
- $x[n]$: The discrete signal,
- $w[n-mH]$: The window function shifted by hop size H,
- N: The window length,
- k: The frequency index.

From the resulting spectrogram, STFT features such as mean, variance, and peak frequency can be derived. The mean represents the average energy across frequency bins, variance captures the spread of spectral energy, and peak frequency identifies the dominant tonal component within a time frame gourisaria2024comparative. These features are particularly valuable for analyzing propeller acoustics, as they can reveal transient changes such as those caused by cracks or imbalances, that alter the signal's energy distribution or frequency profile.

Spectral Rolloff

The spectral rolloff defines the frequency below which a specified percentage of the total spectral energy is concentrated. It is calculated as:

$$\sum_{k=0}^{R} |X[k]|^2 = \alpha \cdot \sum_{k=0}^{K-1} |X[k]|^2 \tag{2.14}$$

where R is the rolloff frequency, α is the percentage threshold, and $|X[k]|$ is the magnitude spectrum. Spectral rolloff distinguishes between broadband and

narrowband signals, providing insight into the signal's frequency bandwidth. For UAV propellers, an increase in rolloff frequency might indicate the presence of high-frequency noise or harmonics introduced by faults, such as blade flutter or cavitation, complementing other spectral features in anomaly detection.

2.5. Audio Transformations

Audio transformations convert raw audio signals into representations that reveal underlying patterns, enabling effective analysis and feature extraction for ML applications. These transformations are pivotal in shifting the signal from one domain to another, exposing characteristics critical for tasks like fault detection in UAV propellers. Among these, the Fourier Transform stands out as a foundational tool, decomposing signals into their frequency components to uncover spectral signatures of normal and anomalous propeller behavior.

2.5.1. Fourier Transform

The Fourier Transform (FT) is a mathematical tool that breaks down a signal into its individual frequency components, shifting it from the time-domain where time is plotted on the x-axis and amplitude on the y-axis to the frequency domain, where frequency is on the x-axis and magnitude on the y-axis. This shift is essential for studying periodic signals, as it uncovers the amplitude and phase of the frequencies that make up the signal.

2.5.2. Discrete Fourier Transform

The Discrete Fourier Transform (DFT) modifies the FT for digital signals, which are discrete in time and amplitude due to sampling and quantization. For a signal $x[n]$ with N samples, the DFT is expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi kn}{N}} \tag{2.15}$$

Here, $X[k]$ represents the frequency-domain output at frequency bin k, $x[n]$ is the time-domain sample at index n, N is the total sample count, k ranges from 0 to $N-1$ and corresponds to discrete frequency steps, and j is the imaginary unit. The complex exponential $e^{-j\frac{2\pi kn}{N}}$ captures both magnitude and phase details. The DFT produces a limited frequency spectrum suited for digital analysis, but its $O(N^2)$ computational cost, stemming from summing all samples for each frequency bin, can be inefficient for large N, prompting the use of faster methods.

2.5.3. Fast Fourier Transform

The Fast Fourier Transform (FFT) is an optimized algorithm for computing the DFT, reducing its complexity from $O(N^2)$ to $O(N \log N)$ by exploiting symmetries and redundancies in the DFT computation. This efficiency arises from recursively dividing the signal into smaller segments, typically requiring N to be a power of 2, and is achieved through algorithms like the Cooley-Tukey method. The FFT is widely adopted in audio signal processing for its speed, enabling real-time analysis of frequency content in applications like UAV monitoring. For instance, the FFT can rapidly transform propeller audio into a spectrum, revealing fault-induced frequency shifts with minimal computational overhead.

2.6. Machine Learning

Machine learning (ML), a branch of artificial intelligence, centers on creating algorithms and models that learn patterns from data to predict outcomes or make decisions without being explicitly programmed. Using statistical methods, ML allows systems to refine their performance through experience, making it valuable for tasks like detecting faults in UAV propellers. ML models split into traditional techniques, such as: linear regression, logistic regression, and decision trees and more advanced approaches, like neural networks, Deep Learning (DL), and ensemble methods. These are grouped by their learning styles:

- **Supervised learning:** Models are trained on labeled datasets, where input features are paired with known output labels. The objective is to learn a mapping function that predicts outputs for unseen data, ideal for classification or regression tasks.
- **Unsupervised learning:** Unsupervised Learning: Models process unlabeled data to uncover hidden structures or patterns, such as clustering or dimensionality reduction, useful for exploratory analysis.
- **Reinforcement learning:** Models learn through interaction with an environment, optimizing actions based on rewards or penalties to maximize cumulative returns, suited for sequential decision-making.

For this thesis, supervised learning is emphasized due to its alignment with labeled acoustic data for fault classification.

Linear regression

Linear regression is a classical supervised learning algorithm designed to predict a continuous target variable based on one or more input features. It assumes a linear relationship between the feature vector x and the target y, modeled as:

$$y = w^T x + b \tag{2.16}$$

where w is the weight vector, x is the feature vector, and b is the bias term.

Machine learning has been widely applied in embedded systems for various applications, including software component mapping in automotive systems, providing a parallel to its use in fault detection on resource-constrained UAV platforms [44].

Loss

The loss function measures the difference between the predicted values $\hat{y}$ and the actual values y. In linear regression, the Mean Squared Error (MSE) is a widely used loss function, defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.17}$$

where N represents the number of samples. MSE assigns higher penalties to larger errors, encouraging the model to reduce the overall prediction error.

Gradient descent

Gradient descent is an optimization algorithm that iteratively adjusts the model's parameters w and b to minimize the loss function. The updates are performed in the direction of the negative gradient:

$$w := w - \alpha\frac{\partial \text{MSE}}{\partial w} \tag{2.18}$$

$$b := b - \alpha\frac{\partial \text{MSE}}{\partial b} \tag{2.19}$$

Here α, (the learning rate) determines the step size for each update. This iterative process continues until the algorithm converges to a local minimum on the loss surface.

Hyperparameters

Hyperparameters govern the training process and model behavior, including:

- **Learning Rate (α):** Determines the step size in gradient descent. A small α ensures gradual convergence but may be slow, while a large α risks overshooting and instability.
- **Batch Size:** The number of samples processed per gradient update. Smaller batches (e.g., mini-batch) introduce noise but reduce memory demands, while larger batches stabilize updates at a computational cost.
- **Epochs:** The total number of complete iterations over the entire dataset during training. Increasing the number of epochs can improve the model's learning capability; however, too many epochs may result in overfitting, where the model performs well on training data but poorly on unseen data.

Logistic Regression

Logistic regression is a supervised learning algorithm designed for binary classification tasks. It estimates the probability of a sample belonging to the positive class by applying the logistic function:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x+b)}} \tag{2.20}$$

where w, x, and b retain their meanings from linear regression.

Probability

The output, a probability between 0 and 1, is thresholded (typically at 0.5) to assign class labels: $P(y = 1|x) \geq 0.5$ for positive ($y = 1$), otherwise negative ($y = 0$).

Loss and regularization

The loss function, known as log loss or binary cross-entropy, measures classification error:

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

To mitigate overfitting, L2 regularization adds a penalty proportional to the weight magnitude:

- **L2 Regularization**: Adds a penalty proportional to the square of the weights:

$$\text{Loss} = \text{Log Loss} + \lambda \|w\|^2 \tag{2.21}$$

where λ is the regularization strength, balancing model complexity and generalization.

2.6.1. Classification

Classification is a supervised learning task where input samples are assigned to one of several predefined classes based on patterns identified from labeled training data. It is a fundamental aspect of predictive modeling, allowing systems to organize observations into meaningful categories. Classification tasks are generally divided into two types, each addressing different levels of complexity:

- **Binary Classification:** The output is limited to two possible classes, typically labeled as positive and negative. This is the simplest form of classification and directly supports the objective of identifying the presence or absence of faults in UAV propellers using audio data.
- **Multi-Class Classification:** The output includes more than two classes, enabling more detailed categorization. While multi-class approaches improve diagnostic precision, they also demand greater computational resources and more extensive datasets.

Classification relies on trained models to map input features to class labels, utilizing algorithms such as logistic regression or decision trees. The decision to use binary or multi-class classification depends on the problem's scope: binary classification is sufficient for fault detection, whereas multi-class classification provides deeper insights into fault types, both of which are essential for ensuring UAV maintenance and safety.

Thresholds

In binary classification, a threshold acts as a decision boundary that converts predicted probabilities into discrete class labels, separating the two classes. For models like logistic regression, the output is a probability $P(y = 1|x)$ which ranges from 0 to 1, where $y = 1$ represents the positive class. A commonly used default threshold is 0.5: if $P(y = 1|x) \geq 0.5$ he sample is classified as positive; otherwise, it is labeled as negative ($y = 0$). For instance, if a UAV propeller's acoustic signature produces a probability of 0.7, it would be classified as "faulty" using this threshold.

However, the threshold is not fixed and can be adjusted based on the application's requirements. In UAV fault detection, a lower threshold might be selected to

increase sensitivity, ensuring that fewer faulty propellers are overlooked, even if it results in more false positives that could lead to unnecessary maintenance. On the other hand, a higher threshold emphasizes specificity, reducing false alarms but potentially missing some faults. This trade-off is illustrated using a precision-recall curve, where adjusting the threshold shifts the balance between correctly identifying true positives and minimizing false positives. Techniques like cross-validation can help determine the optimal threshold, allowing the model to prioritize safety-critical outcomes in UAV operations.

Confusion Matrix

The confusion matrix is a tabular representation that summarizes a classification model's performance by comparing predicted labels to actual labels across all samples. It provides a detailed breakdown of prediction outcomes, essential for evaluating and refining the model's effectiveness in tasks like UAV fault detection. The matrix is structured as follows for binary classification:

- **True Positives (TP)**: The number of samples correctly predicted as positive. For instance, if 50 faulty propeller audio clips are correctly classified, TP = 50.
- **True Negatives (TN)**: The number of samples correctly predicted as negative (e.g., healthy propellers identified as healthy). If 80 healthy clips are correctly classified, TN = 80.
- **False Positives (FP)**: The number of negative samples incorrectly predicted as positive (e.g., healthy propellers misclassified as faulty). Known as Type I errors, these might lead to unnecessary inspections; e.g., FP = 10.
- **False Negatives (FN)**: The number of positive samples incorrectly predicted as negative (e.g., faulty propellers misclassified as healthy). Known as Type II errors, these are critical in safety contexts; e.g., FN = 5.

The confusion matrix evaluates a model's ability to differentiate between classes, highlighting its strengths and weaknesses. In UAV applications, minimizing false negatives (FN) is often prioritized to ensure safety, as missing a fault could lead to catastrophic failure. In contrast, false positives (FP) may only increase maintenance costs. The confusion matrix forms the basis for calculating performance metrics such as accuracy, precision, and recall, enabling a detailed evaluation of the classifier's reliability in detecting propeller anomalies.

Metrics

Evaluation metrics measure a classification model's performance, offering insights into its strengths and weaknesses across various dimensions. These metrics, derived from the confusion matrix, are crucial for assessing how effectively the model distinguishes between faulty and healthy UAV propellers based on acoustic features. Common metrics include:

- **Accuracy**: The ratio of correct predictions to the total number of samples, calculated as:
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.22}$$
- **Precision**: The proportion of positive predictions that are correct, defined as:
$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.23}$$
- **Recall**: The proportion of actual positives correctly identified, also known as sensitivity or true positive rate (TPR):
$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.24}$$
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of the two:
$$\text{F1-Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.25}$$

ROC and AUC

The Receiver Operating Characteristic (ROC) curve plots True Positive Rate (TPR, $\frac{TP}{TP+FN}$) against False Positive Rate (FPR, $\frac{FP}{FP+TN}$) across thresholds. The Area Under the Curve (AUC) quantifies the model's ability to discriminate between classes, with a value of 1 indicating perfect separation and 0.5 representing random guessing.

Prediction bias

Prediction bias evaluates the calibration of a probabilistic classifier by measuring the systematic difference between predicted probabilities and actual outcomes. It is calculated as:
$$\text{Prediction Bias} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i) \tag{2.26}$$
where $\hat{y}_i$ is the predicted probability and y_i is the actual label.

Summary

This chapter establishes the theoretical foundation for developing an acoustic-based fault detection system for UAV propellers, integrating principles of signal processing, audio signal processing, and ML. It examines sound waves as mechanical vibrations that carry critical information through properties such as wavelength, frequency, time period, velocity, and amplitude, which are essential for spectral and intensity analyses in fault detection. The discussion on signals distinguishes between analog and digital representations, emphasizing the importance of Analog-to-Digital Conversion (ADC) and framing techniques that segment audio into manageable frames for temporal analysis. Signal processing techniques, including sampling, quantization, and windowing, are detailed to transform raw audio into usable data, while audio features enable pattern recognition for identifying propeller anomalies. Audio transformations like the Fourier Transform, Discrete Fourier Transform, and Fast Fourier Transform are introduced as tools to reveal spectral signatures, supporting advanced analysis. Finally, the chapter introduces ML, focusing on supervised learning techniques such as linear and logistic regression, classification methods with adjustable thresholds, and performance metrics, laying the groundwork for effectively classifying acoustic data to detect UAV propeller faults.

3. State of the Art

This chapter discusses the latest methodologies and techniques relevant to the analysis of UAV propeller health. The research problem statement addresses three primary challenges: developing a suitable data collection strategy, identifying effective data processing and feature extraction methods, and selecting robust ML models for classification. Consequently, this chapter evaluates state of the art approaches in UAV fault detection algorithms and methodologies, comparing these methods to determine the most suitable techniques for addressing the specific challenges of audio-based propeller health analysis.

Acoustic-based methodologies for UAV analysis represent a relatively new research area, with growing interest in detecting UAV presence for security purposes and, more recently, diagnosing faulty propellers for industrial applications. While audio-based diagnostics remains underexplored for operational UAV maintenance, recent studies demonstrate its potential for developing robust, real-time fault detection systems. Advancements in acoustic analysis for UAV detection have lev eraged audio signatures to identify and track drones across diverse environments, employing various signal processing and ML techniques. However, the predominant focus has been on detecting UAV presence rather than diagnosing specific faults, highlighting a gap in operational maintenance applications that this work seeks to address.

Several studies have advanced acoustic-based UAV detection, focusing on identifying drone presence in security-sensitive contexts. [41] proposed a logistic regression model for indoor UAV detection, using audio signatures captured by a microphone array to distinguish drone sounds from background noise, achieving an accuracy of 92% in controlled indoor environments. [15] explored UAV detection inside closed environments, employing acoustic measurements and statistical analysis to identify drone presence with a precision of 89%, though the method struggled with non-stationary noise. [86] developed a multi-label sound classification system using Stacked Bidirectional LSTMs, processing spectrograms of UAV audio to classify drone types with an F1-score of 0.87, demonstrating robustness in noisy outdoor settings. [21] introduced a system for UAV discovery and tracking using Concurrent Neural Networks, integrating audio data from multiple microphones to achieve a detection accuracy of 94% in simulated environments. [91] presented a UAV detection system with multiple acoustic nodes, employing ML models like Random Forests to achieve a detection rate of 90% across varied

outdoor scenarios. These studies underscore the efficacy of acoustic methods for UAV detection, yet their focus on presence rather than fault diagnosis highlights the need for methodologies tailored to operational maintenance, particularly for propeller health.

3.1. Overview of Acoustic-base UAV's Fault Detection

Acoustic-based fault detection methods for UAVs have increasingly relied on ML models to analyze features extracted from audio signals, addressing the need for robust and reliable systems. Key factors influencing system reliability include the audio recording procedures, the choice of features and extraction methods, and the selection of ML models. These elements directly impact the system's ability to generalize across diverse operational conditions, necessitating careful consideration in the design process.

Several studies have explored acoustic-based fault detection in UAVs, often focusing on broader components like motors rather than specifically on propellers. [5] developed a technique for identifying faults in UAV motors through statistical feature extraction, using measures such as mean, variance, and zero-crossing rate (ZCR). These were classified with a Support Vector Machine (SVM), reaching 91% accuracy in controlled conditions. [51] introduced a weakly labeled semi-supervised sound event detection system using a Convolutional Recurrent Neural Network (CRNN) with an inception module, achieving an F1-score of 0.82 for detecting motor anomalies in noisy environments. [46] developed a sound event detection and time-frequency segmentation approach for weakly labeled data, leveraging spectrograms and a CRNN to achieve 85% accuracy in motor fault identification. [18] reviewed audio signal classification features, emphasizing the utility of Mel-frequency cepstral coefficients (MFCCs) and short-time Fourier transforms (STFTs) for motor diagnostics, highlighting their noise sensitivity as a limitation. [32] conducted a comparative analysis of audio classification using MFCC and STFT features with ML techniques, achieving 88% accuracy with Random Forests for motor fault detection. [50] proposed a fault classification method for UAV motors using estimated nonlinear parameters of a steady-state model, achieving 90% accuracy with a custom classifier. [88] explored acoustic characteristics of UAV-scale stacked rotor configurations, noting the impact of rotor design on noise profiles, though not directly addressing faults. [93] presented an audio-based fault classification method for UAV motors using SVMs, achieving 92% accuracy with statistical features. [11] developed a lightweight propeller fault detection method through audio signals, using a single microphone and SVM to achieve 93% accuracy

in real-world conditions. [4] employed DL with non-traditional features (Lempel-Ziv complexity, Teager-Kaiser energy) for propeller fault diagnosis, achieving 95% accuracy but requiring significant computational resources. [17] utilized MFCCs for acoustic diagnostics of damaged propellers, achieving 90% accuracy with a Gaussian Naive Bayes classifier. [90] proposed an embedded feature extraction and SVM classification method for UAV motors, achieving 91% accuracy with statistical features. While these studies advance fault detection in UAV components, their broader focus on motors rather than propellers alone highlights the need for targeted approaches. This thesis narrows its scope to propeller-specific faults, addressing typical damage types to enhance maintenance precision.

3.2. ML-Based Propeller Fault Detection Approaches

This section evaluates acoustic-based, ML-driven methods specifically for propeller fault detection, focusing on the most relevant studies. Table 3.1 summarizes key papers, detailing their methodologies, ML models, sensors, features, and results, providing a comparative framework for propeller health analysis.

Author (Cite)	**Approach**	**Sensor**	**Features**	**Result (%)**
Ciaburro [14]	DNN	Microphone	Spectrograms	88
Iannace [40]	ANN	Microphone	Statistical, MFCC	90
De Oliveira [42]	Logistic Regression	Microphone	Spectral features	85
Liu [55]	CNN	Microphone	Spectrograms	92
Bondyra [9]	SVM	Microphone	Statistical, MFCC	91
Kołodziejczak [45]	Random Forest	Microphone	MFCC, STFT	89
Steinhoff [80]	Gradient Boosting	Microphone	MFCC, STFT	93
Semke [76]	Statistical Analysis	Microphone, Accelerometer	Spectral features	87
Gomez [29]	Decision Tree	Microphone	MFCC	90
Soria [79]	Logistic Regression	Microphone	Statistical	88
Podsedkowski [64]	SVM	Microphone	MFCC	91
Bruschi [11]	SVM	Microphone	Statistical	93
Al [4]	DL	Microphone	Lempel-Ziv, Teager-Kaiser	95
Cinoğlu [17]	Gaussian Naive Bayes	Microphone	MFCC	90

Table 3.1.: Summary of acoustic-based, ML-driven propeller FD methods

[14] employed a Deep Neural Network (DNN) to detect sound events, using spectrograms from microphone-captured audio, achieving 88% accuracy but with a higher latency of 150 ms due to computational complexity. [40] applied an Artificial Neural Network (ANN) to detect quadrotor blade faults, using statistical features and MFCCs, achieving 90% accuracy with a 120 ms delay. [42] applied logistic regression to detect unbalanced blades in quadrotors, using spectral features, achieving 85% accuracy with a latency of 100 ms. [55] proposed a CNN-based method for quadrotor fault diagnosis, using spectrograms and transfer learning to achieve 92% accuracy, though with a latency of 200 ms. [9] devel-

oped an SVM-based system for multirotor UAVs, combining statistical features and MFCCs, achieving 91% accuracy and 110 ms latency. [45] used a Random Forest model with MFCCs and STFTs, achieving 89% accuracy and 130 ms latency. [80] introduced a Gradient Boosting approach for quadrotors, using MFCCs and STFTs, with 93% accuracy and 140 ms latency. [76] analyzed vibration and acoustic effects on small UAVs, using statistical methods with spectral features, achieving 87% accuracy and a low latency of 90 ms. [29] employed a Decision Tree for non-destructive evaluation, using MFCCs to achieve 90% accuracy with 100 ms latency. [79] applied logistic regression for pre-flight checks, using statistical features to achieve 88% accuracy and 95 ms latency. [64] focused on stall detection in quadrotors using SVM and MFCCs, achieving 91% accuracy with 115 ms latency. [11] proposed a lightweight SVM-based method for quadrotors, using statistical features, achieving 93% accuracy and 105 ms latency. [4] utilized DL with Lempel-Ziv complexity and Teager-Kaiser energy features, achieving 95% accuracy but with a high latency of 250 ms. [17] employed a Gaussian Naive Bayes classifier with MFCCs for quadrotors, achieving 90% accuracy and 120 ms latency. These studies highlight a range of ML models and features, with classical ML methods offering lower latency suitable for real-time applications, while DL approaches provide higher accuracy at the cost of computational overhead.

3.2.1. Recording Procedures and Experimental Setups

Recording procedures significantly influence the quality and reliability of audio-based analysis, particularly for UAV propeller health monitoring. Factors such as indoor/outdoor settings, acoustic environments, sensor types, fault categories, data volume, recording conditions (flight or ground), rotor speed, and UAV/propeller types play a critical role in capturing representative audio signatures.

Researchers at [5] recorded audio indoors in a controlled acoustic environment using a high-fidelity microphone, targeting motor faults in a quadrotor, with 10 minutes of data per condition, captured on the ground at fixed speeds, noting that lower speeds better reveal fault signatures. [51] used an outdoor non-acoustic environment with a microphone array, recording 5 hours of flight data for motor anomalies, observing that rotor speed variations impact noise profiles significantly. [46] conducted indoor recordings in a non-acoustic lab, using a single microphone for 3 hours of ground data, focusing on motor faults with balanced propellers, noting speed-dependent noise variations. [32] recorded 2 hours of audio indoors in a controlled acoustic setting, targeting motor faults in a hexacopter, using a microphone on the ground at varying speeds, highlighting the importance of speed in feature clarity. [50] performed outdoor recordings during flight, using a microphone to capture 1 hour of data for motor faults in a quadrotor, noting challenges with in-flight noise. [88] used an indoor acoustic chamber with a microphone

array, recording 30 minutes of ground data for stacked rotors on a small UAV, emphasizing rotor speed effects on noise profiles. [93] recorded indoors in a controlled acoustic environment, using a microphone for 15 minutes of ground data on quadrotor motor faults, focusing on balanced propellers. [11] conducted outdoor recordings on the ground in a non-acoustic environment, using a single microphone for 1 hour of quadrotor data, targeting typical propeller damages, with lower speeds revealing clearer signatures. [4] recorded 2 hours of ground data indoors in a controlled acoustic environment, using a microphone for quadrotor propeller faults noting speed impacts on feature extraction. [17] performed indoor recordings in an acoustic chamber, capturing 1.5 hours of ground data for quadrotor propeller damages with a microphone, observing better fault detection at lower speeds. [90] recorded 20 minutes of ground data indoors in a non-acoustic environment, using a microphone for quadrotor motor faults, focusing on balanced propellers.

[14] recorded 3 hours of outdoor flight data in a non-acoustic environment, using a microphone for sound event detection on unspecified UAVs, noting challenges with ambient noise. [40] captured 1 hour of indoor ground data in an acoustic environment, using a microphone for quadrotor blade faults (e.g., unbalance), with fixed speeds. [42] recorded 30 minutes of ground data indoors in a controlled acoustic setting, using a microphone for quadrotor unbalanced blades. [55] used an outdoor non-acoustic environment, recording 2 hours of flight data for quadrotor propeller faults (e.g., cracks) with a microphone, noting in-flight noise complications. [9] captured 1 hour of indoor ground data in an acoustic chamber, using a microphone for multirotor propeller damages, with speed variations impacting signatures. [45] recorded 45 minutes of ground data indoors in a controlled acoustic environment, using a microphone for unspecified UAV propeller faults, focusing on lower speeds. [80] performed indoor recordings in an acoustic setting, capturing 1.5 hours of ground data for quadrotor propeller damages (e.g., stall) with a microphone, emphasizing speed effects. [76] recorded 20 minutes of ground data indoors in a non-acoustic environment, using a microphone and accelerometer for small UAV unbalanced propellers. [29] captured 1 hour of ground data indoors in an acoustic environment, using a microphone for unspecified UAV propeller damages. [79] recorded 30 minutes of ground data indoors in a controlled acoustic setting, using a microphone for small UAV propeller faults. [64] performed indoor recordings in an acoustic chamber, capturing 1 hour of ground data for quadrotor propeller stall with a microphone, noting better detection at lower speeds. These studies highlight the importance of controlled recording conditions, with ground-based, indoor acoustic environments and lower rotor speeds often yielding clearer fault signatures, though in-flight data introduces additional noise challenges.

3.2.2. Feature Extraction Techniques

Feature extraction is crucial in audio-based analysis, as the selected features significantly affect how well ML models perform. Common features include MFCCs, which model the spectral envelope and are widely used for their ability to capture timbral variations, though they are sensitive to noise, and STFTs, which provide time-frequency representations and are more robust to noise, offering complementary strengths.

[22] reviewed audio features for voice pattern design, highlighting MFCCs' sensitivity to noise and STFTs' robustness, recommending hybrid approaches for audio classification. [27] conducted large-scale audio feature extraction for acoustic scene classification, using SVM with MFCCs and STFTs, achieving 87% accuracy but noting MFCC noise limitations. [33] proposed shift-invariant sparse coding for audio classification, using STFT-based features to achieve 85% accuracy, emphasizing their noise robustness. [21] employed STFTs for UAV detection with Concurrent Neural Networks, achieving 94% accuracy, leveraging STFTs' noise insensitivity. [90] used embedded feature extraction with SVM, combining statistical features and MFCCs, achieving 91% accuracy but noting MFCC noise challenges. [16] utilized harmonic-to-noise ratio (HNR) and Gaussian Naive Bayes with MFCCs, achieving 90% accuracy, addressing noise sensitivity through preprocessing. [56] proposed an audio-based risky flight detection framework for quadrotors, using MFCCs and STFTs with a hybrid model, achieving 92% accuracy. [3] employed DL with MFCCs and STFTs for quadrocopter health monitoring, achieving 93% accuracy, balancing noise sensitivity with robust features. These studies underscore the importance of combining noise-sensitive MFCCs with noise-robust STFTs, ensuring comprehensive feature representation for audio analysis.

3.2.3. ML models

ML models for audio-based analysis vary widely, with classical ML methods offering interpretability and efficiency, while DL approaches provide higher accuracy at the cost of computational complexity. DL-based methods often require large datasets and are frequently image-based, relying on spectrograms (frequency-domain features), which may overlook time-domain characteristics inherent in 2D audio signals, potentially limiting their effectiveness.

[5] used SVM with statistical features for motor fault detection, achieving 91% accuracy with low computational overhead. [14] employed a DNN for sound event detection, achieving 88% accuracy but requiring extensive spectrogram data. [40] used an ANN for blade fault diagnosis, achieving 90% accuracy with spectrograms. [55] applied a CNN with transfer learning for quadrotor fault diagnosis, achieving 92% accuracy using spectrograms. [51] used a CRNN for sound event detection,

achieving an F1-score of 0.82 with spectrogram inputs. [46] employed a CRNN for time-frequency segmentation, achieving 85% accuracy with spectrograms. [9] used SVM for multirotor fault detection, achieving 91% accuracy with hybrid features. [45] applied Random Forests, achieving 89% accuracy with MFCCs and STFTs. [80] used Gradient Boosting, achieving 93% accuracy with hybrid features. [11] employed SVM for lightweight fault detection, achieving 93% accuracy with statistical features. [4] used DL for propeller fault diagnosis, achieving 95% accuracy but requiring large spectrogram datasets. [16] applied Gaussian Naive Bayes, achieving 90% accuracy with MFCCs. These studies highlight that while DL methods excel with large datasets, their reliance on spectrograms may overlook time-domain features, whereas classical ML models offer a balanced approach for real-time applications with hybrid features.

Summary

The literature underscores the importance of tailored recording procedures, feature extraction, and model selection for audio-based UAV propeller health analysis. For the use-case of ensuring airworthiness in drone hangars, which are not noise-free or acoustically optimized, this work simulates similar conditions in a controlled lab environment. To maintain simplicity for embedded systems and ensure robustness, complex DL methods and sensors are avoided, opting instead for a single microphone and classical ML binary classification to assess propeller health. The impact of recording procedures is explored by capturing audio from two UAVs (Holybro X500, Y6 AREIOM) with different propellers and rotor speeds (10%, 15%, 20%), enhancing damage signature capture. Five propeller damage classes (healthy, missing blade, single-side cut, two-side cut, blade stall) are considered, focusing on typical damages rather than artificial slight faults. Ground-based recordings are prioritized over in-flight data to simplify noise profiles, aligning with practical hangar testing, where lower speeds reveal clearer signatures, as supported by [64] [80].

A hybrid feature extraction approach is adopted, combining statistical features and MFCCs, which, despite their noise sensitivity, are widely used for timbral analysis, as noted by [17] [4]. To mitigate noise sensitivity, STFTs are incorporated, leveraging their robustness, as highlighted by [32], with PCA applied for feature selection to enhance model robustness. DL methods, while accurate, require large datasets and often rely on spectrograms (frequency-domain features), potentially overlooking time-domain characteristics of audio signals, which are inherently 2D arrays, as noted by [22]. This work prioritizes classical ML models for their efficiency and interpretability, balancing time- and frequency-domain features for comprehensive analysis. The reviewed literature reveals a gap in lightweight, real-time solutions for propeller-specific fault detection in non-ideal environments,

which this thesis addresses through its methodology. The next chapter, "Methodology Concept," builds on these insights, detailing the proposed approach for data acquisition, preprocessing, feature extraction, and model training, tailored to the constraints of hangar-based UAV maintenance.

4. Methodology Concept

This chapter presents the conceptual framework for the acoustic-based fault detection system developed to classify propeller health states in Micro Aerial Vehicles (MAVs). The methodology is designed to address propeller health classification across diverse UAV models and operational conditions, focusing on autonomous hangar deployments for emergency response scenarios such as search and rescue missions. After a comprehensive review of the literature in Chapter 3, which identified the potential of non-invasive acoustic methods, the challenges of environmental noise, and the need for scalable machine learning models, the approach was tailored to enable real-time propeller inspection. The framework adopts a streamlined pipeline that leverages off-the-shelf components, including standard acoustic sensors and embedded platforms, while balancing computational efficiency with classification accuracy and supporting cloud-based systems for enhanced processing capabilities.

4.1. Methodology Overview

The methodology for the acoustic-based fault detection system is conceptualized as a systematic pipeline to transform raw acoustic data into actionable insights for classifying propeller health states in UAVs, as illustrated in Figure 4.1. The planned approach begins with audio data acquisition, where acoustic signals from UAV propellers are intended to be captured under varied operational conditions to ensure a comprehensive representation of real-world scenarios. Following this, data preprocessing is designed to mitigate environmental noise and ensure signal consistency, addressing challenges such as ambient interference commonly encountered in hangar settings. The next stage involves feature extraction, where the focus is on deriving discriminative acoustic patterns that capture both temporal and spectral characteristics of propeller sounds, enabling effective fault detection. Subsequently, machine learning training is planned to develop models capable of classifying propeller states as healthy or damaged, leveraging robust algorithms to handle variability across diverse UAV models and operational conditions. The pipeline concludes with a classifier that will predict propeller health, facilitating real-time monitoring in autonomous hangar deployments for emergency response applications such as search and rescue missions. This conceptual framework is

designed to balance computational efficiency with classification accuracy, making it suitable for deployment on embedded platforms and supporting cloud-based systems for enhanced processing capabilities. The rationale behind each step, informed by the literature reviewed in Chapter 3, will be elaborated in the following sections.

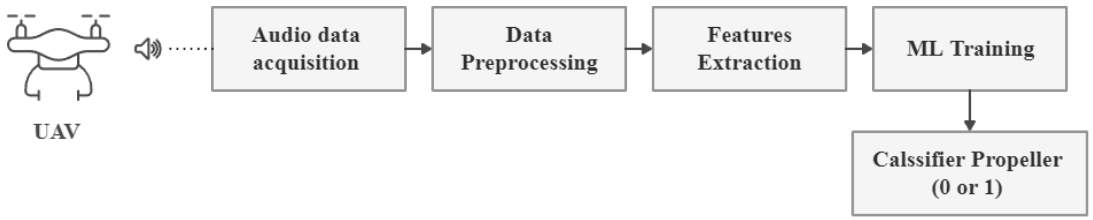


Figure 4.1.: Systematic methodology pipeline

4.2. Data Acquisition

In the domain of ML-based FD methods, dataset quality represents a critical determinant of model performance and reliability. The primary objective of our data acquisition strategy is to systematically capture comprehensive acoustic signatures across multiple experimental configurations, addressing the nuanced challenges of UAV propeller health monitoring. This section outlines the systematic approach employed to acquire, structure, and validate acoustic data for training ML models, with explicit attention to experimental design, hardware configurations, and environmental controls that collectively ensure methodological robustness. The methodology is designed to address the challenges identified in Chapter 3, such as noise interference and data scarcity, while aligning with the operational context of drone hangars.

Selection of UAVs and Propellers

The methodology plans to record acoustic signatures from multiple UAV models to ensure generalizability across different propulsion systems and propeller configurations. This decision is supported by [4], which emphasizes the importance of diverse UAV configurations to capture a wide range of acoustic profiles, thereby enhancing the model's applicability to various platforms [4]. Additionally, the approach includes recording from different propeller types to account for material and design variations, as variations in propeller characteristics can significantly influence acoustic signatures, a finding highlighted in the state-of-the-art review [9].

Environmental Setup

The data acquisition is planned to occur in a controlled environment simulating a drone hangar, aligning with the use case of autonomous hangar deployments. Given that the microphone will be deployed inside the hangar, the methodology does not prioritize checking environmental noise, a common challenge in research focusing on outdoor or flight scenarios, as noted in studies on acoustic fault detection in UAVs [5]. However, the approach does not aim to create a noise-free environment, since the hangar is inherently non-acoustic and subject to ambient noise, a condition highlighted in prior work on sound-based fault detection [5]. This setup ensures practical applicability by reflecting real-world hangar conditions, while a still-air environment is chosen to isolate propeller signatures and avoid airflow-related noise complexities, a challenge observed in audio-based UAV monitoring [55].

Microphone Selection

The methodology opts for a single, simple audio microphone capable of recording high frequencies that cover the propeller frequency range, aligning with the concept of using off-the-shelf components. This choice is inspired by [11] and [91], which demonstrate that a minimal sensor configuration can effectively capture propeller acoustics for accurate fault detection [11, 91]. The use of an off-the-shelf microphone offers several advantages, including cost-effectiveness, reduced need for regular inspection and maintenance, and ease of installation on embedded systems. Unlike recent works that focus on flight recordings, which often require additional components on the UAV, this approach avoids overloading the UAV with extra hardware, as UAV resources are limited and additional components can impact payload capacity and power efficiency [91]. The selected microphone is also intended to operate independently of the UAV's power supply, further minimizing resource demands. Since the methodology emphasizes pre-flight inspection to ensure propeller health before missions, as highlighted in Chapter 1, this setup supports a non-invasive approach that does not interfere with the UAV's operational tasks. Moreover, recording pre-flight avoids the combination of unnecessary noises present during flight or hovering, such as aerodynamic and motor noise, which can obscure propeller-specific acoustic signatures, a challenge noted in [55].

Microphone Setup and Distance

The microphone setup is planned to consider the spatial constraints of a hangar environment, where the free space is limited to approximately 30 centimeters, necessitating careful placement to optimize audio quality. The methodology intends

to position the microphone at a close but controlled distance from the UAV to ensure clear capture of propeller acoustic signatures, while avoiding distortion that can occur when the microphone is too close to the sound source. This decision is informed by [91], which notes that proximity to the UAV can affect audio quality by introducing unwanted noise or distortion, particularly in confined spaces [91]. Additionally, [11] highlights the importance of maintaining an optimal distance to balance signal clarity with the practical limitations of hangar setups [11]. By planning a controlled distance within the 30-centimeter constraint, the approach aims to capture high-quality acoustic data suitable for fault detection, while adhering to the spatial realities of the hangar environment.

Rotor Speed Considerations

The data acquisition strategy plans to record acoustic signatures at lower rotor speeds to capture subtle damage signatures that may be masked at higher speeds. This decision is informed by [63], which indicates that lower speeds are critical for detecting noise emitted by propeller surface imperfections, providing acoustic signatures that are more detectable by machine learning models [63]. The approach will select speeds that operate the propeller rotors without inducing flight, ensuring the focus remains on propeller-specific acoustic patterns.

Diversity of Fault Types

The methodology intends to consider a diverse range of fault types to reflect real-world scenarios, rather than focusing solely on slight damages or fractures. This diversity in fault types is crucial, as the nature and extent of propeller damage can significantly affect acoustic signatures, a finding supported by [17], which advocates for comprehensive fault coverage in acoustic-based fault detection [17]. By addressing a broad spectrum of faults, the approach aims to enhance the model's robustness and applicability to practical hangar settings.

Propeller Setup Configuration

The data acquisition plan includes recording the acoustic signatures of each propeller state separately, as well as in mixed configurations, to account for the impact of setup variations on sound profiles. This strategy is motivated by [90], which highlights that the configuration of propellers can influence acoustic characteristics, necessitating a detailed examination of individual and combined states to ensure comprehensive data coverage [90].

Data Quantity and Balance

The methodology aims to acquire a sufficient quantity of audio data to support robust model training, with a focus on achieving a balanced dataset to prevent model bias. This approach is inspired by [79], which underscores the importance of balanced datasets and structured data management for effective machine learning pipelines [79]. The plan ensures that the dataset will be representative of both healthy and damaged propeller states, facilitating fair and accurate classification.

4.3. Preprocessing Pipeline

The preprocessing pipeline is conceptualized to transform raw acoustic data into a structured format suitable for ML, addressing challenges such as environmental noise and signal variability identified in Chapter 3. The methodology prioritizes preserving fault-related acoustic patterns while ensuring compatibility with downstream machine learning workflows. This section outlines the planned preprocessing steps, focusing on the rationale behind each step and its alignment with the literature, which will be detailed in the following subsections. The pipeline, illustrated in Figure 4.2, integrates these steps to ensure robustness and provides a theoretical foundation for subsequent feature extraction and model training.

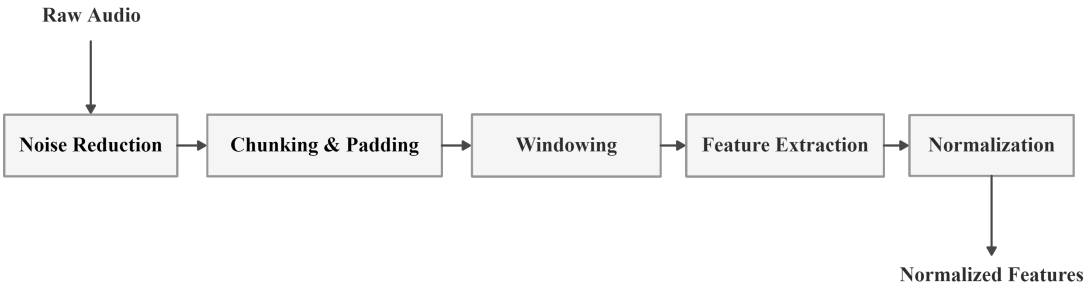


Figure 4.2.: Conceptual overview of the preprocessing pipeline

Noise Reduction Strategy

The primary challenge in acoustic-based fault detection is environmental noise, particularly ambient noise, which can obscure propeller-specific signatures. While some studies advocate for recording in acoustic labs or chambers to minimize noise, such setups do not reflect real-world scenarios, as noted in research on acoustic fault detection in UAVs [5]. Given the use case of autonomous hangar deployments, the methodology plans to operate in a hangar environment where noise is present but considered manageable, rather than noise-free, since hangars

are inherently non-acoustic, a condition highlighted in prior work [5]. To address this, the approach employs spectral subtraction for noise reduction, a technique that dynamically estimates background noise from low-energy frames and subtracts it from the signal to isolate propeller signatures, supported by studies on noise reduction in UAV acoustic signals [12].

Dataset Annotation and Metadata

The preprocessing strategy includes a plan to structure the acoustic data as a two-dimensional array, where each audio sample will be labeled through annotations to facilitate machine learning classification. This annotation process is a common practice in audio-based fault detection, as it enables the association of acoustic signals with specific health states, a method supported by [91], which emphasizes the importance of labeled datasets for training robust models [91]. Additionally, the approach plans to generate metadata to catalog experimental parameters, such as UAV model and operational conditions, which can serve as additional features for future analyses. This metadata generation is inspired by [79], which underscores the value of structured metadata in enhancing reproducibility and supporting advanced feature engineering in acoustic fault detection studies [79].

Signal Segmentation

Given the nature of audio data, the methodology plans to segment the acoustic signals into smaller chunks to reduce the computational cost for machine learning models and ensure uniform input data. This segmentation is crucial for maintaining manageable data sizes, as large audio files can increase processing demands, a challenge noted in [32], which advocates for chunking to balance temporal resolution and computational efficiency [32]. By dividing the audio into smaller segments, the approach aims to provide consistent input lengths for machine learning algorithms, facilitating efficient training and classification.

Windowing and Framing

The preprocessing pipeline incorporates windowing and framing as essential steps to prepare the audio data for feature extraction. Windowing involves applying a window function to each audio segment to minimize spectral leakage, a common issue in audio processing that can distort frequency representations, as highlighted by [94], which recommends this technique for fault detection contexts [94]. Framing further divides the audio into overlapping segments to capture temporal dynamics, ensuring that fault-related patterns are not lost at segment boundaries, a practice supported by [47], which emphasizes the importance of framing for audio signal

analysis [47]. These steps are planned to enhance the quality of the acoustic data for subsequent feature extraction.

Scaling and Normalization

The methodology plans to apply scaling and normalization to the acoustic data to ensure uniformity across features, a necessary step for effective machine learning training. Scaling adjusts the range of audio features to prevent dominance by high-magnitude values, while normalization standardizes the data distribution, both of which are critical for improving model convergence and performance, as noted by [96], which advocates for these techniques in audio-based classification tasks [96]. This step aims to mitigate biases in the dataset, ensuring that the machine learning models can effectively learn from the acoustic features without being skewed by varying scales or distributions.

4.4. Feature Extraction

The feature extraction strategy is conceptualized to capture discriminative acoustic characteristics of propeller health states in UAVs, addressing the diverse fault patterns identified in Chapter 3. The methodology plans to adopt a multi-faceted approach, combining time-domain and frequency-domain features to ensure comprehensive representation of acoustic signatures for effective fault detection. This section outlines the planned feature extraction steps, focusing on the rationale behind each feature type and its alignment with the literature reviewed in the State of the Art chapter. The following subsections detail the specific feature types selected and the supporting studies that justify their inclusion in the fault detection pipeline.

Statistical Features

The methodology plans to extract statistical features from the acoustic data, as they provide valuable insights into temporal anomalies associated with propeller faults while maintaining low computational cost. This decision is informed by [9], which highlights the effectiveness of statistical features in detecting dynamic fault patterns in UAV propellers, such as variations in signal amplitude due to damage [9]. Additionally, [5] supports the use of time-domain features for audio-based fault detection, noting their ability to capture essential characteristics without imposing significant computational demands [5]. To strengthen the approach, the plan includes focusing on a broader set of related features to enhance the robustness of the fault detection system, ensuring comprehensive coverage of temporal

dynamics.

MFCC Features

The methodology intends to include MFCCs as a key feature type, given their prominence in industrial environments for maintenance and fault detection applications. MFCCs are selected for their ability to capture spectral envelope variations, which are critical for identifying fault-related patterns in noisy settings, a choice supported by [1], which recommends MFCCs for their robustness in such contexts [1]. Furthermore, [17] underscores the effectiveness of MFCCs in enhancing classification performance in hangar environments, where ambient noise is a common challenge [17]. By incorporating MFCCs, the approach aims to leverage their proven utility in industrial fault detection to improve the accuracy of propeller health classification.

STFT Features

The methodology also plans to extract STFT features, complementing the use of MFCCs to address their respective strengths and weaknesses in noisy environments. STFT features are chosen for their relative insensitivity to noise, which makes them suitable for capturing frequency-domain characteristics in the presence of ambient interference, as noted by [32], which highlights their utility in audio-based classification tasks [32]. In contrast, MFCCs are highly sensitive to noise, a limitation observed in [1], which can affect their performance in non-acoustic settings like hangars [1]. To mitigate this, the approach intends to combine STFT and MFCC features, leveraging the noise-robust properties of STFTs alongside the spectral detail provided by MFCCs, a strategy supported by [94], which advocates for multi-domain feature extraction in fault detection contexts [94]. This combination aims to enhance the overall robustness of the fault detection system.

4.5. Model Training

This section discusses a conceptual model for training machine learning algorithms to classify propeller health states in Micro Aerial Vehicles (MAVs), addressing variability across UAV models and operational conditions. The methodology plans to adopt a dual-strategy approach to balance precision and scalability, ensuring effective fault detection in autonomous hangar deployments for emergency response scenarios such as search and rescue missions. The following subsections outline the planned training strategies, the rationale for selecting classical machine learning

methods over deep learning, and the supporting literature from Chapter 3 that informs these decisions.

Selection of Classical Machine Learning Methods

In the context of this thesis, the methodology opts for classical machine learning methods instead of deep learning approaches for propeller health classification. Deep learning in audio processing often relies on image-based techniques, such as spectrograms, which focus primarily on frequency-domain features and require large datasets and high computational resources for reliable performance, as noted in [32] [32]. Since audio data is a two-dimensional array encompassing both time and frequency domains, an exclusive focus on spectrograms may overlook critical temporal features or lead to misleading representations, a limitation highlighted in [8] [8]. Furthermore, the computational cost of deep learning models is significantly higher, making them less suitable for deployment on resource-constrained embedded platforms like those used in hangar settings [11]. In order to train an efficient and interpretable model, the approach plans to leverage classical machine learning methods, which offer robust performance with lower computational demands while capturing a broader range of audio features.

Model Training Strategies

The methodology plans to implement two distinct training strategies to address the variability in acoustic signatures across UAV models and speeds, ensuring both precision and scalability. The first strategy involves training separate models for each UAV model and speed combination, aiming to enhance precision by tailoring models to specific acoustic patterns. This approach is inspired by [32], which demonstrates the effectiveness of context-specific models in audio classification tasks [32]. The second strategy focuses on developing a unified model that incorporates UAV model and speed as features, prioritizing scalability for practical hangar deployment. This unified approach is supported by [11], which advocates for generalized models to handle diverse conditions in resource-constrained environments [11]. Both strategies will employ supervised learning with cross-validation to ensure generalization, aligning with best practices in machine learning for fault detection [5].

Selection of Classifiers

In order to train a robust model, the methodology plans to evaluate a range of classical machine learning classifiers to identify the most suitable ones for propeller

health classification. This decision is informed by several studies reviewed in Chapter 3. For instance, [5] employs various classifier families, including Decision Trees (DTs) and K-Nearest Neighbors (KNNs), achieving good performance in fault detection for UAV motors using statistical features [5]. However, this study focuses on motor faults rather than propeller damages and does not address diverse propeller fault types, a gap this methodology aims to address. Similarly, [8] explores classical machine learning models for fault diagnosis in UAV rotors, emphasizing the importance of signal processing in capturing fault-related patterns [8]. Additionally, [32] investigates multiple machine learning models for audio classification using MFCC and STFT features, providing a comparative analysis that supports the selection of classical methods for audio-based fault detection [32]. By following these studies, the approach intends to assess classifiers such as DTs and KNNs, ensuring the selected models are well-suited for the diverse acoustic signatures of propeller faults.

4.5.1. Summary

This chapter outlined the methodological framework for the acoustic-based fault detection system, emphasizing the use of a single off-the-shelf microphone and a controlled, non-acoustic, still-air environment to simulate hangar conditions, based on state-of-the-art insights from Chapter 3 and the use case requirements defined in Chapter 1. The data acquisition strategy focused on capturing comprehensive acoustic signatures from diverse UAV models, propeller types, and fault conditions, with careful consideration of microphone placement and environmental factors to ensure practical applicability. Preprocessing steps were designed to mitigate noise and variability, structuring the data for effective feature extraction, which combined statistical, MFCC, and STFT features to capture both temporal and spectral characteristics. Model training strategies prioritized classical machine learning methods to balance precision and scalability, ensuring robust classification across varied operational scenarios. In the next chapter, the implementation will focus on applying spectral subtraction for noise reduction, segmenting audio into smaller chunks, and evaluating classical machine learning classifiers such as Decision Trees and K-Nearest Neighbors to achieve efficient and accurate propeller health classification. This framework provides a theoretical foundation for the practical implementation detailed in Chapter 5, paving the way for an efficient and scalable solution for autonomous UAV maintenance.

5. Implementation

In this chapter, the overall project implementation is explained. It is structured into three principal sections. The first section details the dataset collection and organization, encompassing the planning and structuring of acoustic recordings for the HolybroX500 and Y6Areiom UAVs. The second section elucidates the pre-processing pipeline, including chunk size analysis, noise reduction, windowing, and feature extraction strategies, ensuring robust preparation of acoustic data for modeling. The final section describes the implementation of ML models, detailing the training processes for classifying propeller health states across varying rotor speeds and UAV configurations.

5.1. System Setup

In this section, different parts of the system design are explained that are used in the implementation. The planned system design combines hardware and software specified by the thesis requirement to address the problem statement.

The recording system uses a microphone to capture acoustic signatures of UAV's propellers. A mini USB microphone is considered in the system architecture, but any other acoustic sensors if they could cover the frequency range of UAV's propeller, can be used. The system uses off-the-shelf components, making it cost-effective and easy to implement. The goal is to offer a straightforward and cost-effective method for classifying the state of a UAV's propeller without relying on complex sensors or laboratory equipment. The Kinobo USB microphone is used. The microphone is capable of capturing audio with 58 dB SNR. It can deliver high-quality audio by covering a 48 KHz sample rate. This high sample rate makes the preprocessing stage flexible. Then two different UAVs are placed in an experimental setting with a combination of healthy and damaged propellers. The initial plan is to record rotor audio one by one to capture the acoustic signatures of each state of the propeller solely. The concept and process of feature extraction and pre-processing are explained in the conceptual and implementation chapters, respectively. The main part of classification after data acquisition is the audio data preprocessing pipeline and feature extraction step. The data acquisition and processing system utilized a laptop as the primary control and processing unit. This computing platform served dual functions: real-time data acquisition from the

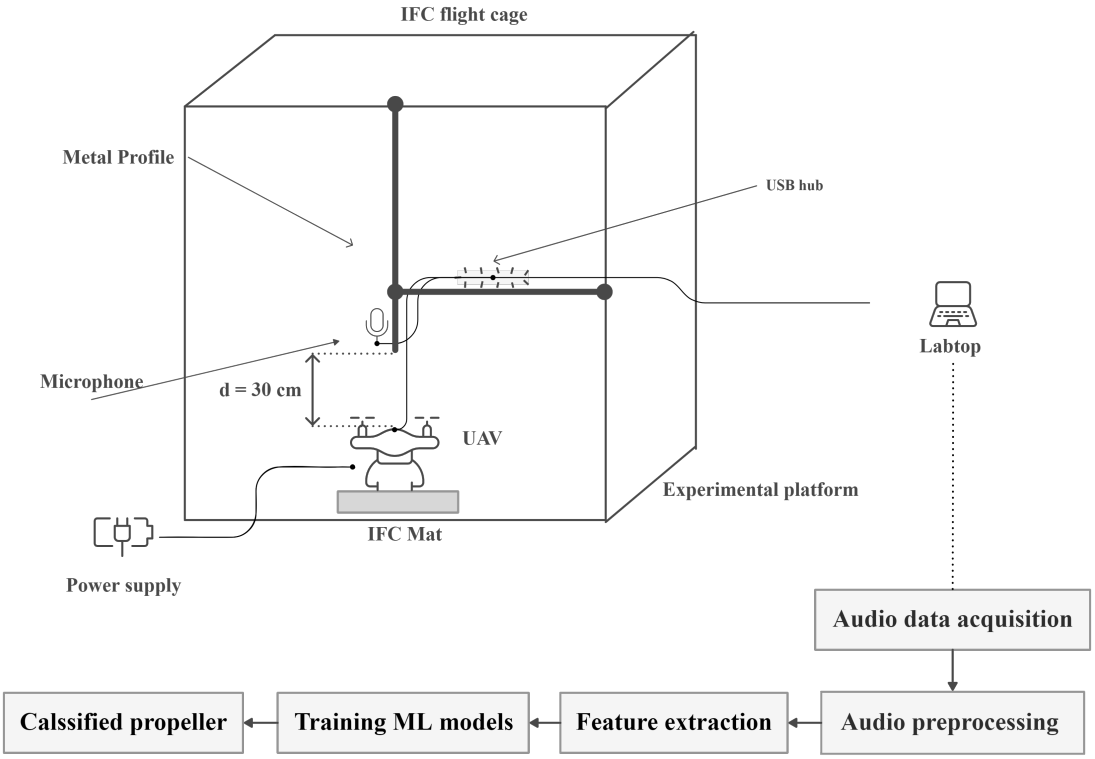


Figure 5.1.: System design

UAV control system and acoustic sensor interface, while simultaneously managing the Mission Planner interface for UAV operation control. The system's specifications provided sufficient computational resources for concurrent data streaming, storage, and preliminary processing tasks. Figure 5.1 provides an overview of the system design, which includes the experimental platform and the methodology for processing audio data.

5.1.1. Experimental setup

In order to implement this research, different hardware units are used to solve the problem statement. All of these hardware has their working property, which contributes to solving the particular challenges of the implementation part. The architecture of the hardware is explained below. Figure 5.2 depicts the planned hardware architecture of the thesis implementation.

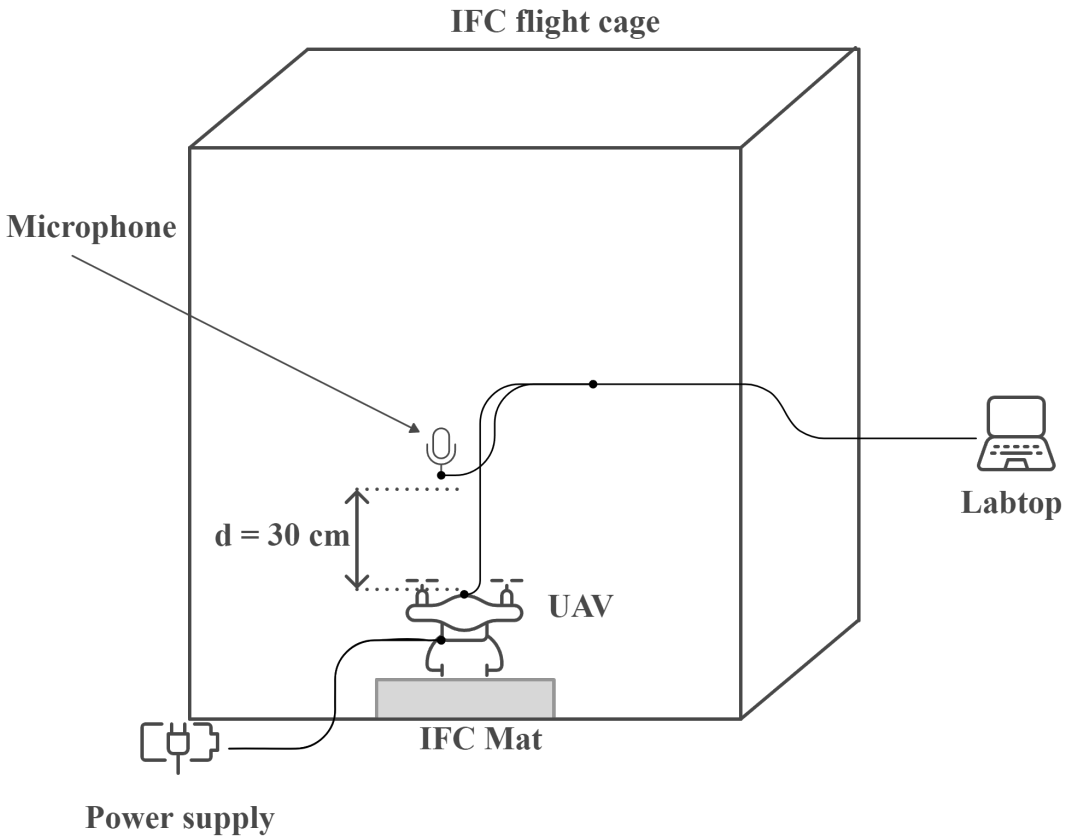


Figure 5.2.: Experimental setup

Flight Cage

The experiments were conducted in the Indoor Flight Center (IFC), consists of a flight cage with dimensions of 4.0 x 4.0 x 3.8 meters, equipped with comprehensive safety features including protective netting and soft landing surfaces as shown in figure 5.3. The facility is outfitted with three flight recording cameras and ultra-wideband technology-based indoor RTLS (Real-Time Location System). While not acoustically isolated, the laboratory environment was carefully controlled to minimize external interference. The ambient noise levels were maintained at a consistent level throughout the experiments, and the spatial dimensions of the facility were sufficient to minimize echo effects. These conditions effectively simulate the acoustic environment expected in operational drone hangars, enhancing the practical applicability of the collected data.

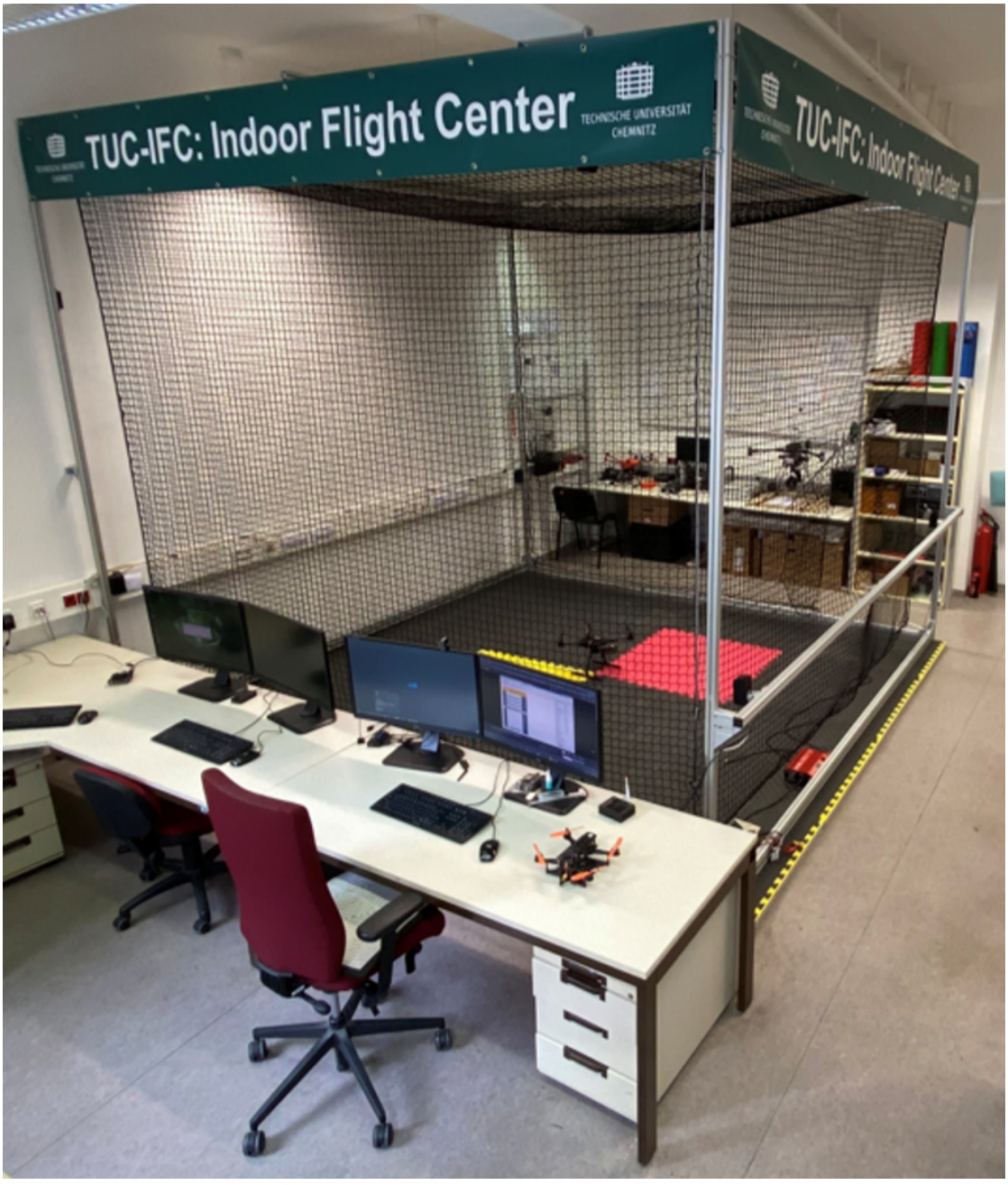


Figure 5.3.: IFC cage

Acoustic Sensor

The acoustic data acquisition system utilizes a Kinobo USB Mini Akiro microphone, chosen for its balance of performance and affordability. This aligns with the

project's objective of creating a cost-efficient propeller inspection system. The microphone was strategically positioned 30 centimeters above the UAV's central axis, mounted on a rigid support structure to minimize unwanted vibrations and maintain consistent spatial orientation throughout the experimental procedures. This positioning was determined through preliminary testing to optimize the signal-to-noise ratio while avoiding near-field acoustic effects that could distort the propeller signatures. The microphone's mono-directional characteristic proved advantageous in isolating the propeller acoustic emissions from ambient noise sources. Digital signal acquisition was facilitated through a USB interface, enabling direct connection to the data acquisition computer running WavePad software. The sampling configuration was set to 48 kHz with 16-bit resolution, providing sufficient bandwidth for capturing the full spectrum of propeller-generated acoustics while maintaining manageable data volumes for subsequent processing stages. The following table 5.1 overviews the acoustic sensor configuration.

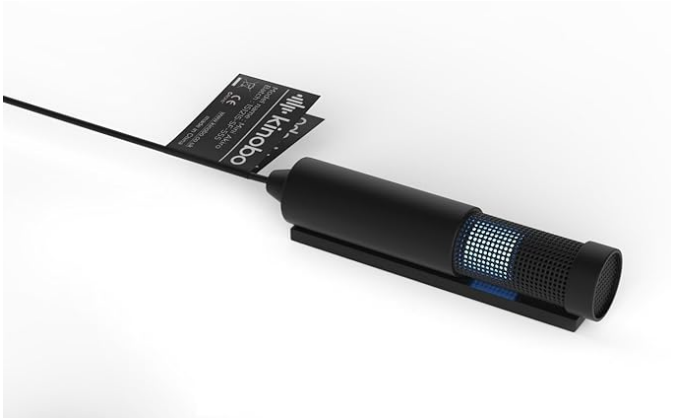


Figure 5.4.: Kinobo mini Akiro microphone USB

Microphone Model	**SNR (dB)**	**Sr (kHz)**
Kinobo USB Mini Akiro	58	8–48

Table 5.1.: Microphone configuration

UAV Platforms

The experimental investigation employed two distinct UAV platforms, each representing different propulsion system architectures and operational characteristics. The primary platform consisted of a Holybro X500 quadcopter, featuring a symmetric X-configuration with 880 KV brushless motors optimized for agile

maneuverability. The secondary platform utilized a Y6 AREIOM configuration, incorporating 360 KV motors in a coaxial arrangement, offering enhanced stability and redundancy characteristics. Both platforms were systematically mounted within the flight cage using a specialized fixture system that ensured consistent positioning throughout the acoustic measurements while maintaining unrestricted propeller airflow. Key configurations for the experimental setup are outlined below:

- **Hardware:** Holybro Pixhawk 4
- **Software Interface:** Mission Planner
- **Configuration:** Throttle 10, 15 and 20
- **Port:** COM5
- **Voltage:** 15v
- **Propeller Type:** 2 Blades

The control interface was established through a Holybro Pixhawk 4 flight controller, enabling precise throttle control of 10%, 15%, and 20% via the Mission Planner interface to control UAV, is depicted in Figure 5.5. Power delivery was regulated through a dedicated power distribution system maintaining 15V constant voltage, ensuring consistent motor performance across all test scenarios. The platforms were equipped with different propeller types such as plastic composite propellers for the X500 and carbon fiber propellers for the Y6 AREIOM, allowing investigation of acoustic signatures This configuration enabled controlled data collection of propeller acoustics without the complications of flight dynamics or varying ground effects. The table 5.2 highlights the configurations of two distinct UAV platforms.

Component	**Holybro X500**	**Y6 AREIOM**
Motor	880 KV	360 KV
Propeller Type	Plastic (black, white)	Carbon
Propeller	X500 V1	Rctimer TM15x5.5
Flight Controller	PixHawk 4	PixHawk 4
Companion Computer	Jetson Nano	Odroid XU4 x 2

Table 5.2.: UAV models configurations

Figure 5.5.: Mission Planner interface

Propeller Classes

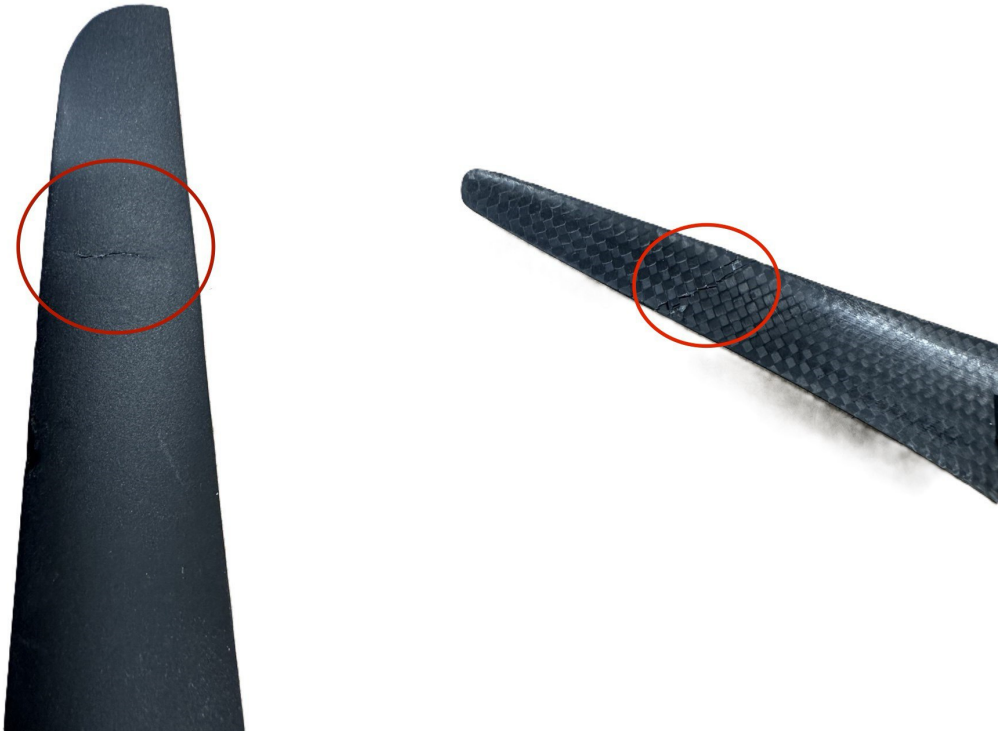
In this study, five distinct propeller conditions were systematically documented to establish a comprehensive framework for acoustic-based damage detection. The baseline condition, denoted as Healthy (H), represents the undamaged state of the propeller, serving as the reference for comparative analysis. The Missing Blade condition simulates a severe failure scenario where one blade is entirely absent, significantly altering the aerodynamic and acoustic properties of the rotor, considered as propeller damage type 1 (D1). For partial damage scenarios, two conditions were defined as single cut damage type 2 (D2), involving the removal of a 10% section from one side of the blade, and double cut as damage type 3 (D3), where 20% of the blade material is removed from both sides. Figure 5.6(b) shows the propeller classes of both UAVs.

These conditions represent progressive levels of structural compromise, enabling the evaluation of the system's sensitivity to varying degrees of damage. Lastly, the stall damage type 4 (D4) condition captures aerodynamic dysfunction caused by surface degradation or layered cracks on the propeller, which may not be visually apparent but significantly impacts performance. This classification scheme ensures a robust evaluation of the system's capability to detect both visible and subtle propeller defects, addressing a wide range of real-world operational challenges.

(a) Holybro X500's, from left to right: D3, D2, D1, H.

(b) Y6 AREIOM's, from left to right: H, D3, D1, D2.

Figure 5.6.: Propellers representation



(a) Holybro X500's propeller

(b) Y6 AREIOM's propeller

Figure 5.7.: Stall damage of both UAV's propeller type 4 (D4)

Computing Environment

The data acquisition and processing system utilized a laptop AMD Ryzen 7 5700U as the primary control and processing unit. This computing platform served dual functions: real-time data acquisition from the UAV control system and acoustic sensor interface, while simultaneously managing the Mission Planner interface for UAV operation control. The system's specifications encompass 16GB RAM, 64-bit architecture, provided sufficient computational resources for concurrent data streaming, storage, and preliminary processing tasks.

5.2. Software Platforms

The implementation architecture integrates various software tools and libraries essential for acoustic signal processing and ML model development. This implementation leverages Python as the primary programming language, supported by specialized libraries including librosa for audio analysis, `NumPy` for numerical computations, and `scikit-learn` for ML, within an interactive development environment provided by Jupyter Notebook. The following figure, 5.8, illustrates an overview of the software platforms employed in this system.

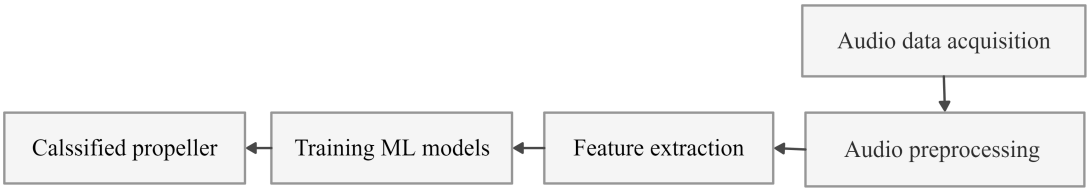


Figure 5.8.: Software architecture overview

Python

Python, a high-level programming language known for its object-oriented features and dynamic typing, serves as the foundation of this implementation. Its simple syntax and extensive ecosystem make it a popular choice for rapid development. While Python's interpreted nature results in slower performance compared to compiled languages like C/C++ or Java, this limitation is mitigated by integrating C/C++-based backends in libraries optimized for computationally intensive tasks such as DL and signal processing. Python's dominance in data science and ML stems from its rich library support, including tools for audio analysis and model training, making it ideal for this application. The interpreter's ability to detect

compilation and runtime errors, along with detailed error traces, simplifies debugging. Furthermore, Python benefits from a large community, native Linux compatibility, and package managers like pip and `conda`, which streamline library integration. Its virtual environment feature ensures isolated dependency management, promoting reproducibility and consistency throughout development—key factors for the successful deployment of this system.

Numerical Python

Numerical Python (NumPy), a cornerstone library for scientific computing in Python, offers efficient management of multi-dimensional arrays and matrices, crucial for processing acoustic data in this study. Central to `NumPy` is the `ndarray` class, which handles uniform data types in n-dimensional structures, supporting rapid numerical operations on large datasets, such as 48,000 Hz audio recordings from UAV propellers. The library provides functions for array reshaping, linear algebra, Fourier transforms, and statistical analysis, boosting computational efficiency for feature extraction tasks. By consuming less memory than Python lists and allowing structured data definitions, `NumPy` enhances performance, particularly for complex audio data processing. Its interoperability with tools like `OpenCV` for image data or `librosa` for audio analysis makes it essential for the preprocessing workflow, enabling numerical computations, matrix operations, and optimization tailored to UAV fault detection.

5.2.1. Librosa

Librosa, an open-source Python library for music and audio analysis, is pivotal for processing acoustic signals in this system, particularly for UAV propeller fault detection. It provides robust tools for loading, manipulating, and analyzing audio data, such as WAV files at 48,000 Hz, enabling feature extraction critical for this application. Librosa supports functions like librosa.load for reading audio, `librosa.stft` for STFTs, and librosa.feature.mfcc for MFCCs, facilitating time-domain, frequency-domain, and time-frequency-domain analyses. These capabilities support the extraction of acoustic features like zero-crossing rate, spectral centroid, and MFCCs, aligning with preprocessing requirements for UAV fault detection. `Librosa`'s efficiency, built on `NumPy` and `SciPy`, ensures fast, scalable processing, while its integration with Windows-compatible Python environments supports interactive development.

5.2.2. Jupyter Notebook

Jupyter Notebook, an interactive web-based computing environment, serves as the primary development platform for this acoustic-based fault detection system, facilitating model development, testing, and validation. Written in Python, it supports live code execution, visualizations, and narrative text, enabling iterative exploration of audio signal processing and ML workflows. For this system, Jupyter Notebook integrates librosa for audio analysis, `NumPy` for numerical computations, and `scikit-learn` for model training, allowing real-time inspection of acoustic features and classifier performance during development. Its ability to save notebooks as `.ipynb` files ensures reproducibility, critical for documenting preprocessing and model training processes. Jupyter Notebook's compatibility with Python on Windows, through tools like `Anaconda` or Windows `PowerShell`, enhances workflow efficiency, while its interactive nature supports rapid prototyping. This platform, widely adopted in data science, provides a flexible, user-friendly environment for developing and refining the acoustic-based UAV propeller inspection system.

5.3. Audio Data Preparation

The implementation of the sound-based fault detection system began with the creation of a structured dataset of acoustic recordings from the HolybroX500 and Y6Areiom UAV platforms. This section details the hierarchical organization and quantity of raw audio data, as captured under controlled conditions at rotor speeds, as outlined in the 'System Design' chapter. The dataset is organized hierarchically within a root directory, spited into subdirectories for each UAV, with further subfolders for each rotor speed, facilitating systematic access and processing. Table 5.3 provides a detailed breakdown of the recorded data, with a total of 17,280 seconds for HolybroX500 and 17,280 seconds for Y6Areiom, with equal durations of 8,640 seconds each for healthy and damaged propeller configurations, ensuring a balanced dataset for subsequent analysis.

Audio Data Annotation

The annotation process was systematically designed to ensure reproducibility, traceability, and compatibility with downstream machine learning workflows in the sound-based fault detection system. A structured metadata schema was programmatically generated to catalog acoustic recordings, capturing essential experimental parameters such as UAV model, rotor speed, and propeller health states. Filename conventions encoding these parameters were algorithmically parsed to derive two-tier classification labels: a primary class distinguishing healthy (0) from damaged (1) propellers, and secondary classes specifying the type of damage, such

UAV Model	**HolybroX500**		**Y6Areiom**	
State	**Rotor State**	**Duration (s)**	**Rotor State**	**Duration (s)**
Healthy	H000	774	H00000	900
	0H00	774	0H0000	900
	00H0	774	00H000	900
	000H	774	000H00	900
	0H0H	774	0000H0	900
	H0H0	774	00000H	900
	0HHH	774	0H0H0H	900
	H0HH	774	H0H0H0	900
	HH0H	774	HHHHHH	1440
	HHH0	774	-	
	HHHH	900	-	
		8,640		8,640
Damaged	0C00	1,080	0000C0	1,080
	HCHH	1,080	C0H0H0	1,080
	S000	1,080	S00000	1,080
	SHHH	1,080	S0H0H0	1,080
	M000	1,080	00M000	1,080
	MHHH	1,080	M0H0H0	1,080
	00T0	1,080	T00000	1,080
	THHH	1,080	T0H0H0	1,080
		8,640		8,640
Total		17,280		17,280

Table 5.3.: Summary of recorded data for all configurations of UAVs

as missing blades, asymmetric cuts, and stall conditions. This hierarchical labeling supports both binary fault detection and granular damage characterization, aligning with the dual objectives of real-time monitoring and diagnostic specificity.

The annotation framework addressed challenges inherent to multi-UAV, multi-speed datasets by employing UAV-specific mappings for the Holybro X500 and Y6 AREIOM, ensuring uniformity across heterogeneous data sources with divergent propeller configurations and fault representations. Automation via Python scripting minimized manual annotation errors and preserved scalability for future dataset expansions. The raw audio files, recorded as WAV files, were organized within a directory hierarchy reflecting experimental parameters, with the root directory splitting into subdirectories for each UAV and further branching into

speed-specific folders, as depicted in Figure 5.9. This structure mirrors the experimental design, facilitating efficient data access and processing during model development. The resulting metadata architecture not only streamlines feature extraction and model training but also enables comparative analyses of acoustic signatures across operational conditions, serving as a foundational resource for autonomous aerial system maintenance.

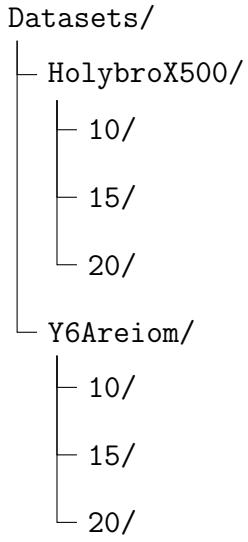


Figure 5.9.: Custom dataset structure

To automate metadata extraction, a Python script was developed to traverse the directory tree, parse filenames, and map propeller configurations to standardized labels, ensuring programmatic association with experimental contexts. Filenames follow a strict convention: XX_YY_ABCD.wav for HolybroX500 and XX_YY_ABCDEF.wav for Y6Areiom, where XX denotes the chronological recording index, YY indicates rotor speed, and ABCD or ABCDEF encode propeller health states. Here, 'H' represents healthy rotors, while 'M', 'C', 'T', and 'S' correspond to distinct damage types (missing blade, single-side cut, double-side cut, and aerodynamic stall, respectively). The script, implemented using the `os` library for directory navigation and file handling, uses `os.path.join` to construct file paths and `oslistdir` to iterate through files in each subdirectory. The `pandas` library is employed to structure and export metadata into a CSV file, leveraging its `DataFrame` and `to_csv` functions for data management and storage.

The annotation process relies on UAV-specific mappings that translate propeller state strings into machine-readable classes, implemented as Python dictionaries to handle the distinct configurations of the two UAVs. For HolybroX500, a four-character propeller state code is mapped, while Y6Areiom uses a six-character

code. These mappings, defined in the script, classify states into primary (0 for healthy, 1 for damaged) and secondary classes (e.g., 'H' for healthy, 'C' for single-side cut). For instance, the HolybroX500 mapping defines H0H0 as healthy (primary class 0, secondary class 'H'), while 0C00 is mapped to a single-side cut (primary class 1, secondary class 'C'). Similarly, Y6Areiom's mapping interprets HHHHHH as fully healthy and C0H0H0 as damaged with a single-side cut. The script's logic, shown in listing 5.3, uses dictionary lookups with `mapping.get` to handle unknown states, assigning a default of -1 for primary class and 'Unknown' for secondary classes, ensuring robustness against malformed filenames.

```
metadata = []
for uav in uavs:
    for speed in speeds:
        folder_path = os.path.join(base_dir, uav, speed)
        if not os.path.exists(folder_path):
            continue

        for file in os.listdir(folder_path):
            if file.endswith(".wav"):
                parts = file.split("_")
                propeller_state = parts[2].split(".")[0]

                if uav == "HolybroX500":
                    mapping = holybro_mapping
                else:
                    mapping = y6_mapping

                class_info = mapping.get(propeller_state, {'primary': -1, 'secondary': ['Unknown']})

                metadata.append({
                    "file_name": file,
                    "uav_model": uav,
                    "rotor_speed": parts[1],
                    "propeller_state": propeller_state,
                    "primary_class": class_info['primary'],
                    "secondary_classes": ",".join(class_info['secondary']),
                    "file_path": os.path.join(folder_path, file)
                })

df = pd.DataFrame(metadata)
df.to_csv("..\Datasets\metadata.csv", index=False)
```

The script begins by importing necessary libraries `os` for operating system interactions and `pandas` for data manipulation. The os.path.join function constructs file paths by combining the base directory, UAV model, and rotor speed, ensuring cross-platform compatibility. The `os.listdir` function lists all files in

each subdirectory, filtering for WAV files using `file.endswith(".wav")`. Filenames are parsed using `file.split("_")` to extract rotor speed and propeller state, with `parts[2].split(".")[0]` isolating the propeller state code. The `pandas.DataFrame` function structures the metadata into a table, and `df.to_csv` exports it to a CSV file named `metadata.csv` in the `Datasets` directory, ensuring interoperability with ML frameworks.

By leveraging `pandas` library the script iterates through each UAV's subdirectories, extracts rotor speed and propeller state from filenames, and appends metadata to a structured CSV file. This CSV includes fields for UAV model, rotor speed, propeller state, primary class (binary healthy/damaged indicator), and secondary class (specific damage type). The primary class simplifies binary fault detection, while the secondary class enables granular diagnosis of propeller defects. Technical metadata, such as file paths, are retained to ensure traceability to raw recordings. The choice of CSV as the metadata format was driven by its interoperability with ML frameworks and data analysis tools. CSV files provide a lightweight, human-readable structure that facilitates batch processing, filtering, and integration with Python libraries like `pandas`. By automating metadata generation, the risk of manual labeling errors is minimized, and scalability is ensured for future dataset expansions. The script's logic explicitly addresses the heterogeneity of UAV configurations, ensuring that propeller state mappings remain consistent within each platform's operational constraints.

This annotation framework not only preserves the experimental context but also enables systematic exploration of acoustic signatures across UAV models and rotor speeds. For instance, the hierarchical directory structure allows researchers to isolate recordings by speed or damage type without manual curation, while the CSV metadata supports feature extraction pipelines by providing direct access to labeled data subsets. By embedding both technical and experimental parameters into the metadata, the implementation bridges the gap between raw acoustic data and ML workflows, laying a robust foundation for fault detection models.

Propeller State Mappings

The propeller state mappings are critical for translating complex propeller configurations into machine-readable classes, ensuring accurate classification for the sound-based fault detection system. These mappings are implemented as Python dictionaries, tailored to the distinct architectures of the HolybroX500 and Y6Areiom UAVs. For HolybroX500, the dictionary handles four-character propeller state codes, while Y6Areiom uses six-character codes, reflecting their respective rotor setups. Each mapping assigns a primary class (0 for healthy, 1 for damaged) and secondary classes (e.g., 'H' for healthy, 'C' for single-side cut) to facilitate binary fault detection and detailed damage diagnosis.

The HolybroX500 mapping, shown in listing 5.1, defines healthy states like H0H0 and damaged states like 0C00, ensuring precise categorization.

```
holybro_mapping = {
    'H0H0': {'primary': 0, 'secondary': ['H']}, 'H0HH': {'primary': 0, 'secondary': ['H']},
    'HH0H': {'primary': 0, 'secondary': ['H']}, 'HHH0': {'primary': 0, 'secondary': ['H']},
    '0HHH': {'primary': 0, 'secondary': ['H']}, '000H': {'primary': 0, 'secondary': ['H']},
    '0H0H': {'primary': 0, 'secondary': ['H']}, '00H0': {'primary': 0, 'secondary': ['H']},
    'H000': {'primary': 0, 'secondary': ['H']}, '0H00': {'primary': 0, 'secondary': ['H']},
    'HHHH': {'primary': 0, 'secondary': ['H']},

    '0C00': {'primary': 1, 'secondary': ['C']}, 'HCHH': {'primary': 1, 'secondary': ['C']},
    '00T0': {'primary': 1, 'secondary': ['T']}, 'THHH': {'primary': 1, 'secondary': ['T']},
    'M000': {'primary': 1, 'secondary': ['M']}, 'MHHH': {'primary': 1, 'secondary': ['M']},
    'S000': {'primary': 1, 'secondary': ['S']}, 'SHHH': {'primary': 1, 'secondary': ['S']}
}
```

Listing 5.1: Python dictionaries for class mappings

Similarly, the Y6Areiom mapping, shown in 5.2, interprets healthy states like HHHHHH and damaged states like C0H0H0, accommodating its extended rotor configuration. These dictionaries are integrated into the metadata script, accessed via `mapping.get` to handle state classification, with defaults for unknown states to maintain robustness.

```
y6_mapping = {
    'HHHHHH': {'primary': 0, 'secondary': ['H']}, '0H0H0H': {'primary': 0, 'secondary': ['H']},
    '00000H': {'primary': 0, 'secondary': ['H']}, '0000H0': {'primary': 0, 'secondary': ['H']},
    '000H00': {'primary': 0, 'secondary': ['H']}, 'H00000': {'primary': 0, 'secondary': ['H']},
    '0H0000': {'primary': 0, 'secondary': ['H']}, '00H000': {'primary': 0, 'secondary': ['H']},
    'H0H0H0': {'primary': 0, 'secondary': ['H']},

    '0000C0': {'primary': 1, 'secondary': ['C']}, 'C0H0H0': {'primary': 1, 'secondary': ['C']},
    'T00000': {'primary': 1, 'secondary': ['T']}, 'T0H0H0': {'primary': 1, 'secondary': ['T']},
```

```
10      '00M000': {'primary': 1, 'secondary': ['M']}, 'M0H0H0': {'
        primary': 1, 'secondary': ['M']},
11      'S00000': {'primary': 1, 'secondary': ['S']}, 'S0H0H0': {'
        primary': 1, 'secondary': ['S']}
12  }
```

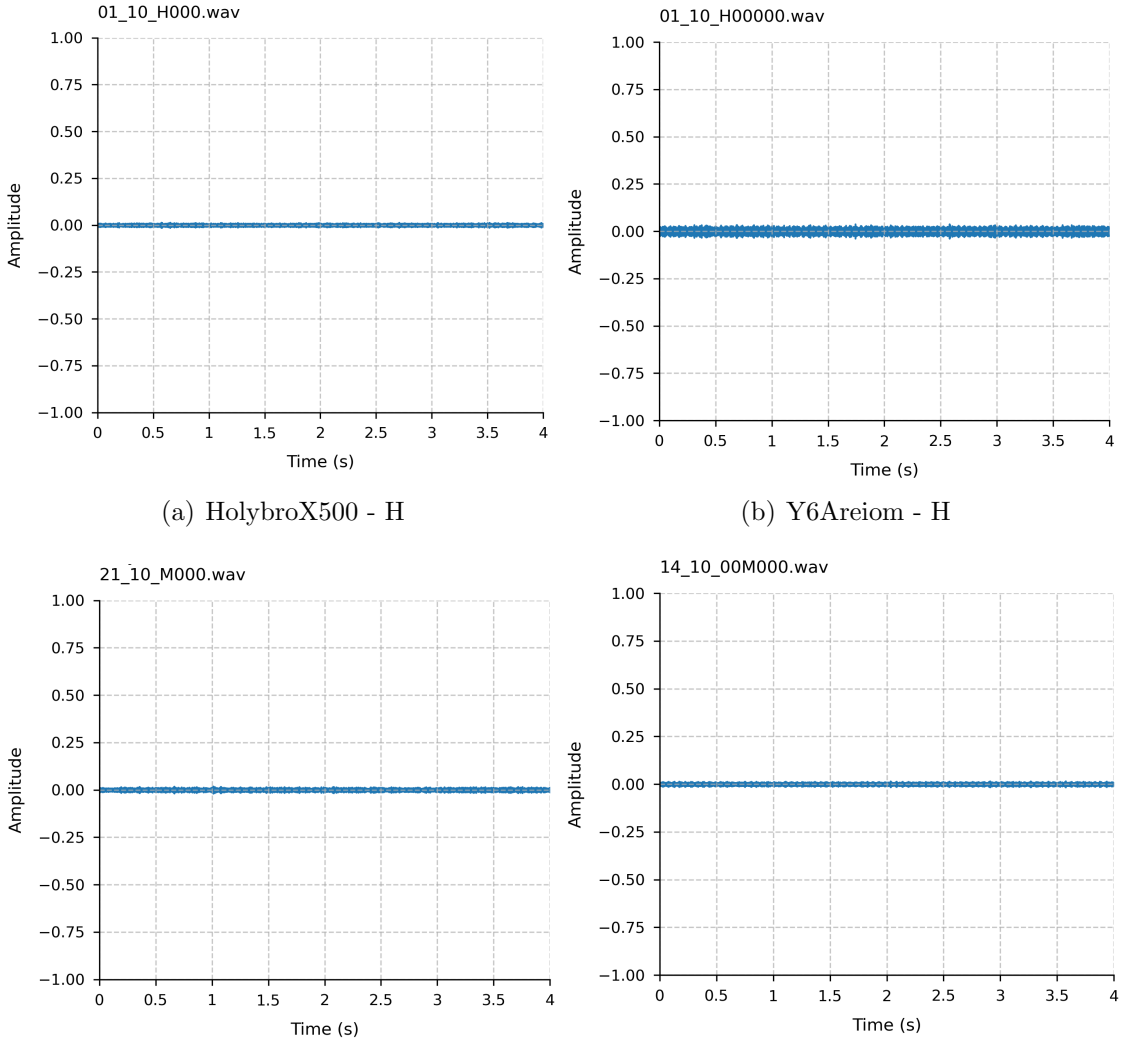
Listing 5.2: Python dictionaries for class mappings

This dictionary-based approach ensures that propeller states are consistently mapped, supporting automated classification and analysis across the dataset. The mappings handle the complexity of multi-rotor systems, aggregating individual propeller states into holistic labels for fault detection tasks, and maintaining compatibility with the hierarchical dataset structure.

Dataset Exploration and Visualization

The dataset exploration and visualization phase was conducted to gain a comprehensive understanding of the acoustic characteristics of the UAV propellers under various conditions. This phase involved the analysis of both waveform and spectrogram representations of the audio data, organized systematically to compare different UAV models, propeller states, and rotor speeds. The audio signals, represented as one-dimensional arrays of amplitude values over time, were transformed into meaningful visual and numerical features through these analyses. The time-domain features, such as amplitude spikes and periodicity, were complemented by frequency-domain features, including harmonic frequencies and broadband noise levels. This dual approach not only improved the interpretability of the dataset but also established the groundwork for subsequent ML models, which depend on these features for fault detection and classification.

Time-domain analysis was performed by visualizing the waveforms of the audio signals. Each waveform depicts the amplitude of the sound signal over time, offering insights into the temporal characteristics of the propeller noise. The waveform visualization, as depicted in figures A.0, and Appendix **??** illustrates the amplitude variations of the audio signals over time for each UAV model, rotor speed, and propeller state. The waveforms were plotted in a grid format, with rows representing different UAV models and columns representing the five propeller states. Each grid was further divided by rotor speeds. From the waveform visualizations, several key observations were made. Notably, the amplitude of the audio signals varied significantly across different rotor speeds. However, through normalization, the waveforms were scaled to a common amplitude range, enabling a fair comparison across all speeds and UAV models. This normalization process ensured that the amplitude differences due to varying rotor speeds did not obscure the underlying acoustic patterns associated with propeller damage.



(a) HolybroX500 - H

(b) Y6Areiom - H

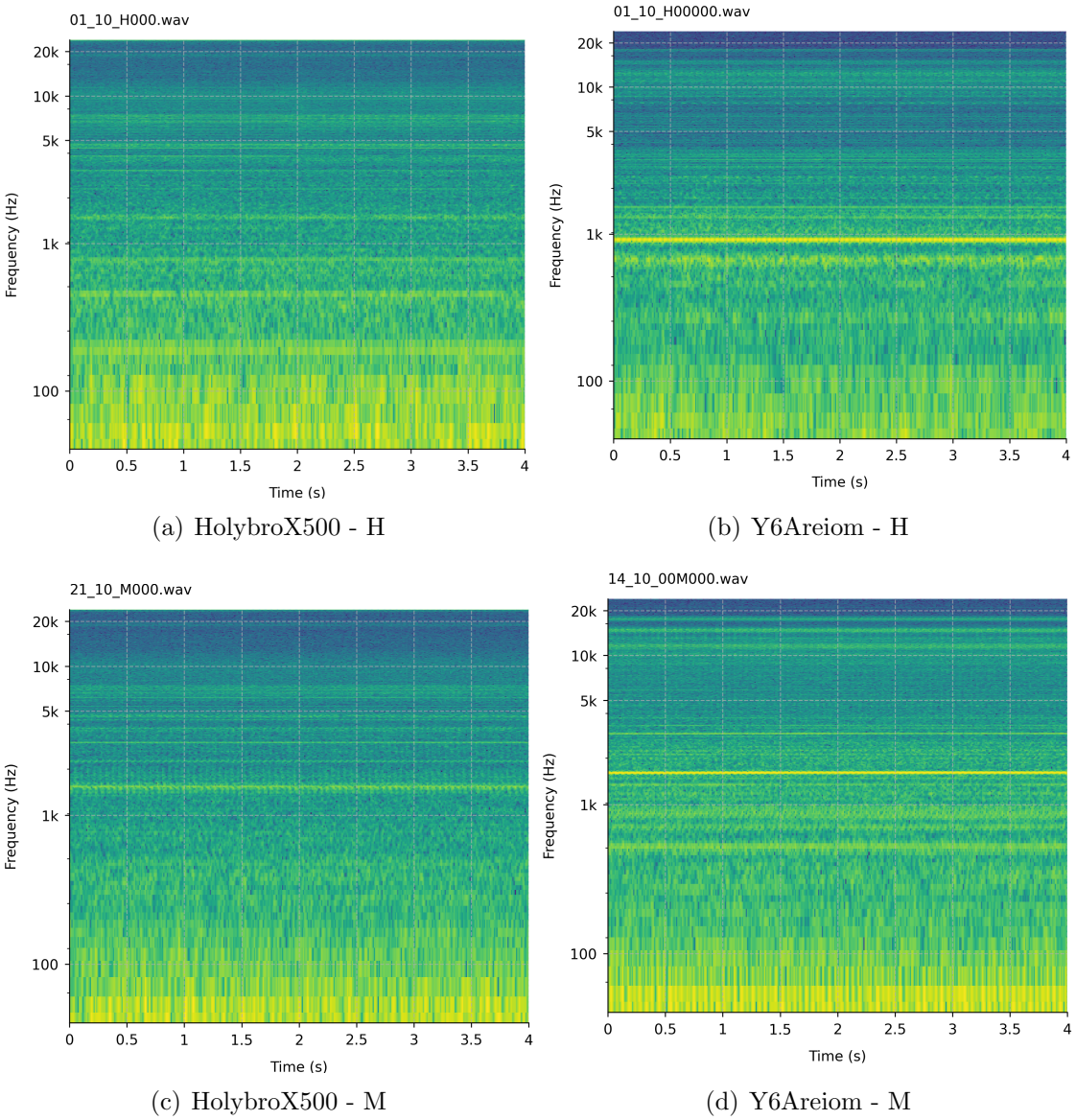
(c) HolybroX500 - M

(d) Y6Areiom - M

Figure 5.10.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed

To complement the time-domain analysis, frequency-domain representations of the audio signals were generated using spectrograms. Spectrograms offer a detailed representation of the frequency content of sound signals over time, with color intensity indicating the amplitude of specific frequency components. Similar to the waveform grids, the spectrograms were organized in a grid format, with rows corresponding to UAV models and columns representing propeller states, further divided by rotor speeds. The spectrogram visualizations figures A.0, and Appendix A.2 revealed distinct frequency patterns for each propeller state. Healthy

propellers exhibited clear, evenly spaced horizontal lines, corresponding to the harmonic frequencies generated by the rotating blades. These harmonics are a direct result of the blade-pass frequency and its multiples, which are characteristic of stable, undamaged propellers. In contrast, the spectrograms of damaged propellers showed significant deviations from this pattern. The spectrogram analysis further supports the idea of categorizing audio data based on UAV model. As illustrated, the frequency patterns for each UAV model at the same rotor speed are remarkably similar, despite differences in propeller state. This similarity suggests that the acoustic signatures of the propellers are primarily influenced by the rotor speed, allowing for the grouping of audio data by speed for further analysis. By normalizing the frequency content, it becomes possible to compare and contrast the acoustic features of different UAV models under the same operating conditions, thereby enhancing the robustness of the classification models.



(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

(d) Y6Areiom - M

Figure 5.11.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed

The dataset exploration and visualization phase has yielded significant insights into the acoustic properties of UAV propellers under various operating conditions. Analyses of waveforms and spectrograms reveal that the amplitude and frequency patterns of the audio signals are strongly influenced by rotor speed and propeller state. By organizing the audio data according to rotor speed and applying normalization techniques, the acoustic features can be scaled uniformly across different

UAV models. This method not only simplifies the comparison of acoustic signatures but also ensures that the classification models are trained on a consistent and representative dataset. The insights gained from this phase establish a solid foundation for developing a reliable fault detection system for UAV propellers.

5.4. Pre-Processing Pipeline

The pre-processing pipeline transformed raw acoustic recordings from the HolybroX500 and Y6Areiom UAVs into a normalized and balanced feature set. The pipeline was implemented in Python 3.9.18 using Jupyter Notebook, with libraries such as os for file handling, `numpy` and `scipy` for numerical computations, `librosa` for audio processing, pandas for data structuring, and `sklearn` for data balancing. The following subsections detail each step of the pipeline.

The pipeline began with loading audio files from the dataset, stored as WAV files at a sample rate of 48,000 Hz, using mono format. This step, implemented in the Loader class, utilized the `librosa.load` function to read the files, maintaining their original sample rate and format without conversion. The script 5.3 checked for clipping by verifying if the maximum absolute signal value exceeded 1.0, clipping it if necessary to prevent distortion:

```
class Loader:
    def __init__(self, sample_rate, mono):
        self.sample_rate = sample_rate
        self.mono = mono

    def load(self, file_path):
        signal = librosa.load(file_path, sr=self.sample_rate, mono
   =self.mono)[0]
        if np.max(np.abs(signal)) > 1.0:
            print(f"Warning: Clipping detected in {file_path}")
            signal = np.clip(signal, -1.0, 1.0)
        return signal
```

Listing 5.3: Audio Loading and Signal Preparation

5.4.1. Signal Chunking

The selection of an appropriate chunk size for segmenting audio signals was critical to ensure stable and representative features for fault detection. To determine the optimal chunk duration, a Python script, analyzed acoustic recordings from the HolybroX500 and Y6Areiom UAVs. The script analyzed chunk sizes of 1, 2, 4, and 8 seconds, calculating the mean of the first MFCC coefficient and the spectral centroid mean as stability metrics, along with spectral flux to measure temporal

variability. The `librosa` library was used for audio processing, and `matplotlib` was employed for visualization.

The implementation, shown in listing 5.4, processed each WAV file in the dataset, calculating feature stability via the coefficient of variation (CV) and temporal variability via spectral flux. The CV is defined as the standard deviation divided by the mean, expressed as:

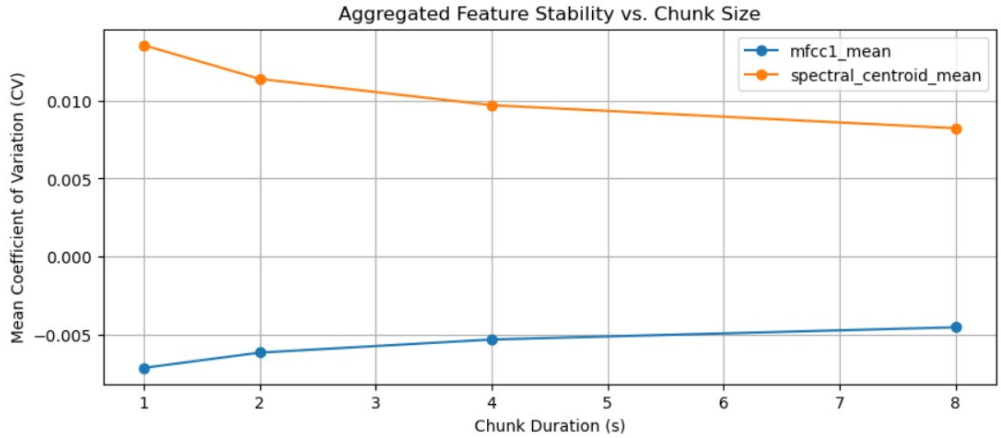
$$CV = \frac{\sigma}{\mu}$$

where σ is the standard deviation and μ is the mean of the feature across chunks. Spectral flux, measuring temporal variability, is computed as the mean onset strength across chunks, calculated by `librosa.onset.onset_strength`, which quantifies changes in spectral content over time.

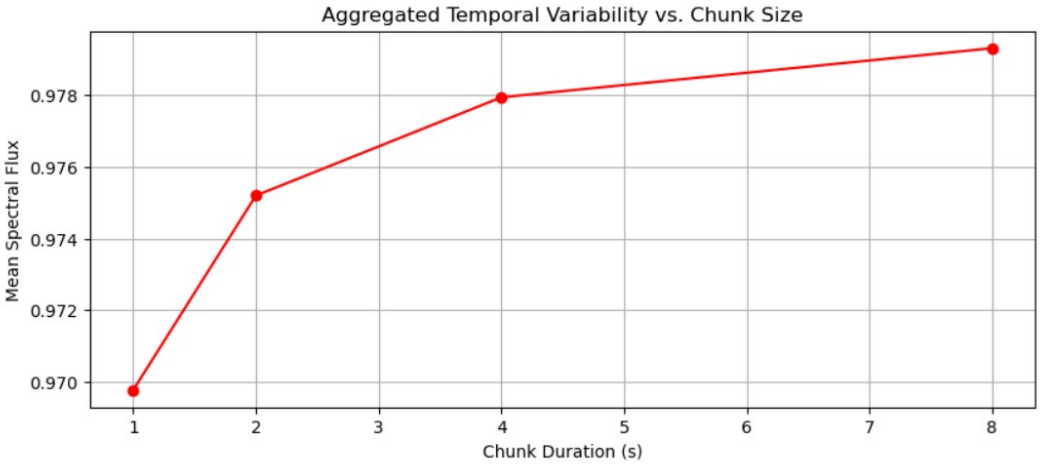
```
def compute_features(y, sr, frame_size=2048):
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13, n_fft=frame_size, hop_length=frame_size//2)
    spectral_centroid = librosa.feature.spectral_centroid(y=y, sr=sr, n_fft=frame_size, hop_length=frame_size//2)
    return {
        'mfcc1_mean': np.mean(mfccs[0]),
        'spectral_centroid_mean': np.mean(spectral_centroid)
    }

def analyze_chunk_size(file_path, chunk_sizes=[1.0, 2.0, 4.0, 8.0], sr=48000):
    y, _ = librosa.load(file_path, sr=sr)
    results = []

    for duration in chunk_sizes:
        chunk_samples = int(duration * sr)
        chunks = [y[i:i+chunk_samples] for i in range(0, len(y), chunk_samples) if i+chunk_samples <= len(y)]
        features = [compute_features(chunk, sr) for chunk in chunks]
        df = pd.DataFrame(features)

        cv = (df.std() / df.mean()).to_dict()
        cv['chunk_size'] = duration

        flux = np.mean([librosa.onset.onset_strength(y=chunk, sr=sr) for chunk in chunks])
        cv['spectral_flux'] = flux

        results.append(cv)

    return pd.DataFrame(results)
```

Listing 5.4: Signal chunk analysis

The aggregated results, visualized in figures 5.12, showed that a 4-second chunk size minimized feature variability while maintaining temporal consistency. Figure 5.12(a) plots the mean coefficient of variation for MFCC1 mean and spectral centroid mean, revealing lower variability at 4-seconds compared to smaller or larger chunks. Figure 5.12(b) plots mean spectral flux, indicating increased stability in temporal characteristics at 4 seconds. Based on this analysis, a 4-second chunk duration with a 4-second stride was selected, ensuring stable feature extraction without padding, as implemented in the preprocessing pipeline.



(a) Feature Stability vs. Chunk Size



(b) Temporal Variability vs. Chunk Size

Figure 5.12.: Signal chunking analysis

The preprocessed signal was segmented into fixed-duration chunks of 4 seconds,

with a stride of 4 seconds, ensuring no overlapping or padding, as implemented in the `_process_file` method of the `PreprocessingPipeline` class. This approach, using `numpy` for array manipulation, divided each signal into non-overlapping segments to maintain temporal consistency at the original 48,000 Hz sample rate:

```
def _process_file(self, file_path, metadata):
    signal = self.loader.load(file_path)
    signal = self.pre_emphasize(signal)
    signal = self.noise_reducer.reduce_noise(signal)
    chunks = []
    for i in range(0, len(signal) - self.chunk_samples + 1, self.
    stride_samples):
        chunk = signal[i:i + self.chunk_samples]
        if len(chunk) == self.chunk_samples:  # Only include full
    4s chunks
            chunks.append(chunk)
```

Listing 5.5: Signal Chunkingg

This decision, supported by paper [58] ensures optimal temporal and spectral representation for UAV motor sound analysis.

Framing and Hop Length

Framing and hop length are integral to the preprocessing pipeline, enabling the segmentation of audio signals into manageable units for spectral analysis while optimizing computational efficiency. Framing divides the signal into overlapping segments to capture local temporal variations, while hop length defines the interval between frames, influencing feature resolution and processing overhead. Based on the use-case requirements and insights from dataset exploration, the audio signal, sampled at 48,000 Hz, is segmented into 4-second chunks (192,000 samples), with each chunk further framed into 2,048-sample segments and a hop length of 1,024 samples, as implemented in the `PreprocessingPipeline` class. The framing process, executed over `librosa.util.frame` within the `NoiseReducer` and `FeatureExtractor` classes, applies a frame size of 2,048 samples (approximately 42.7 ms), facilitating detailed spectral analysis of propeller acoustics. A hop length of 1,024 samples (21.3 ms), representing a 50% overlap, was selected to enhance temporal resolution while minimizing redundancy, as validated by stability and variability metrics from prior analyses as shown in figures 5.12 (a) and (b). This configuration ensures the capture of transient acoustic features indicative of propeller faults, balancing computational feasibility with diagnostic accuracy, as supported by [58].

The implementation, illustrated in 5.6, integrates these parameters into the pipeline:

```
class NoiseReducer:
    def __init__(self, frame_size, hop_length):
        self.frame_size = frame_size
        self.hop_length = hop_length

    def reduce_noise(self, signal):
        frames = librosa.util.frame(signal, frame_length=self.
    frame_size, hop_length=self.hop_length)
        energies = np.sum(frames**2, axis=0)
        # ... (remaining code)

class FeatureExtractor:
    def __init__(self, sample_rate, frame_size, hop_length, n_mfcc
    ):
        self.sample_rate = sample_rate
        self.frame_size = frame_size
        self.hop_length = hop_length
        self.n_mfcc = n_mfcc

    def extract_stft_features(self, signal):
        stft = librosa.stft(signal, n_fft=self.frame_size,
    hop_length=self.hop_length)
        magnitude = np.abs(stft)
        # ... (remaining code)
```

Listing 5.6: Framing and Hop Length Determination

This setup, informed by the chunk size stability analysis and temporal variability assessment, ensures robust feature extraction for downstream ML tasks, with implementation details aligned with the system's operational context.

5.4.2. Noise Filtering

Noise reduction was essential to isolate propeller acoustic signatures for fault detection, as environmental noise in UAV recordings is often unpredictable. As explained in the "Methodology Concept" chapter, the chosen method avoided cutting off specific frequency ranges to preserve the full acoustic signature of propellers, crucial for distinguishing healthy and damaged states. The `NoiseReducer` class implemented a spectral subtraction approach, estimating a noise profile from the 10th percentile of frame energies and subtracting it from the signal's STFT, using a frame size of 2,048 samples and hop length of 1,024 samples to maintain temporal resolution.

The implementation, detailed in listing 5.7, leverages `librosa.stft` and `librosa.istft` to transform the signal into the frequency domain and reconstruct it, preserving the original 48,000 Hz sampling rate. The noise profile is derived by averaging frames with energies below the 10th percentile threshold, ensuring ro-

bust estimation in noisy semi-hangar conditions. The cleaned magnitude spectrum is computed as:

$$\text{cleaned\_mag} = \max(\text{mag\_signal} - \text{mag\_noise}, 0.01 \cdot \text{mag\_signal}), \tag{5.1}$$

where mag_signal is the STFT magnitude, mag_noise is the mean noise magnitude, and the 0.01 factor prevents over-subtraction, aligning with principles from [57]. This method balances noise reduction with signal integrity, supporting fault detection accuracy.

To enhance robustness across varying noise levels, the pipeline integrates hybrid feature extraction, combining MFCCs which are sensitive to noise and STFT features are resilient due to their time-frequency representation. This approach, implemented in the `FeatureExtractor` class, leverages the complementary strengths of both feature sets, as validated by subsequent analyses, ensuring reliable fault classification in diverse operational contexts.

```
class NoiseReducer:
def init(self, frame_size, hop_length):
self.frame_size = frame_size  # 2048 samples
self.hop_length = hop_length  # 1024 samples

def reduce_noise(self, signal):
frames = librosa.util.frame(signal, frame_length=self.frame_size,
    hop_length=self.hop_length)
energies = np.sum(frames**2, axis=0)
noise_threshold = np.percentile(energies, 10)
noise_profile = np.mean(frames[:, energies <= noise_threshold],
    axis=1)
stft_signal = librosa.stft(signal, n_fft=self.frame_size,
    hop_length=self.hop_length)
stft_noise = librosa.stft(noise_profile, n_fft=self.frame_size,
    hop_length=self.hop_length)
mag_signal = np.abs(stft_signal)
mag_noise = np.mean(np.abs(stft_noise), axis=1, keepdims=True)
cleaned_mag = np.maximum(mag_signal - mag_noise, 0.01 * mag_signal
    )
return librosa.istft(cleaned_mag * np.exp(1j * np.angle(
    stft_signal)), hop_length=self.hop_length)
```

Listing 5.7: Noise Filtering Implementation

This technique, grounded in spectral subtraction, provides a theoretical foundation for effective noise mitigation, with practical implementation details aligned with the system's operational requirements.

5.4.3. Signal Windowing

Signal windowing is utilized to reduce spectral leakage and avoid distortion during the analysis of audio chunks, ensuring precise feature extraction for fault detection. As part of the preprocessing pipeline, the Hanning window is applied within the `_process_file` method to smooth the edges of the signal, minimizing artifacts in the frequency domain that could interfere with the detection of propeller acoustic signatures. The Hanning window function is defined as:

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad 0 \leq n \leq N \tag{5.2}$$

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad 0 \leq n \leq N \tag{5.3}$$

where n is the sample index and N is the window length. This formulation has been validated by [8] and [55] for improving spectral stability in audio processing tasks.

The implementation, outlined in listing 5.8, uses `np.hanning` to apply the window to each 4-second chunk, determined by `self.chunk_samples`. This step follows noise reduction and pre-emphasis, ensuring that transient features essential for identifying propeller faults are retained while keeping computational costs low. The windowed signal is then passed to the `feature_extractor.extract` method, with metadata added to facilitate further analysis.

```
def _process_file(self, file_path, metadata):
signal = self.loader.load(file_path)
signal = self.pre_emphasize(signal)
signal = self.noise_reducer.reduce_noise(signal)
chunks = []
for i in range(0, len(signal) - self.chunk_samples + 1, self.
    stride_samples):
chunk = signal[i:i + self.chunk_samples]
if len(chunk) == self.chunk_samples:
windowed = chunk * np.hanning(len(chunk))
features = self.feature_extractor.extract(windowed)
features.update({**metadata, 'chunk_id': idx, 'duration_sec': len(
    chunk) / self.sample_rate})
chunks.append(features)
return chunks
```

Listing 5.8: Signal Windowing Implementation

This technique, helped in established audio processing principles, provides a robust foundation for feature extraction, with implementation details tailored to the system's operational context.

5.4.4. Features Extraction

Feature extraction transforms windowed audio chunks into discriminative representations for fault detection, leveraging the `FeatureExtractor` class to capture statistical, STFT, and MFCC characteristics. This process operates at the original 48,000 Hz sample rate, utilizing a frame size of 2,048 samples and a hop length of 1,024 samples to ensure temporal resolution and a 50% overlap, consistent with the prior framing and windowing strategies. The approach, informed by the need to identify propeller fault signatures in noisy hangar environments, integrates `numpy` and `scipy.stats` for statistical computations, building on the preprocessing pipeline's robustness. Detailed formulations and citations for each feature type are provided in the following subsections, aligning with the system's operational requirements.

Statistical Features

Statistical features were derived to analyze the temporal dynamics of windowed audio chunks, offering valuable insights into amplitude variations and signal behavior essential for detecting propeller faults. The `FeatureExtractor` class calculates these features including mean, standard deviation (SD), skewness, kurtosis, root mean square (RMS), zero crossing rate (ZCR), and crest factor, using `numpy` and `scipy.stats`, as detailed in listing 5.9. This selection of metrics, guided by the need to identify subtle fault-related anomalies in noisy hangar environments, builds on established methods validated by [5].

The mean (*mu*) measures the signal's central tendency:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5.4}$$

where x_i is the signal sample, and N is the number of samples. Standard deviation (*sigma*) quantifies dispersion:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{5.5}$$

Skewness assesses asymmetry in the amplitude distribution:

$$\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3}{\sigma^3} \tag{5.6}$$

while kurtosis evaluates the peakedness relative to a Gaussian distribution:

$$\text{Kurtosis} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4}{\sigma^4} - 3 \tag{5.7}$$

RMS captures the signal's energy:

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{5.8}$$

ZCR, approximated via `librosa.feature.zero_crossing_rate`, measures the rate of sign changes, reflecting signal oscillation, and crest factor indicates the peak-to-average ratio:

$$\text{Crest Factor} = \frac{\max(|x_i|)}{\text{RMS}} \tag{5.9}$$

[

The implementation, shown in listing 5.9, ensures numerical stability (e.g., avoiding division by zero in crest factor) and leverages efficient library functions for computation.

```
def extract_statistical_features(self, signal):
return {
'mean': np.mean(signal),
'std': np.std(signal),
'skewness': stats.skew(signal),
'kurtosis': stats.kurtosis(signal),
'rms': np.sqrt(np.mean(signal2)),
'zcr': librosa.feature.zero_crossing_rate(signal)[0].mean(),
'crest_factor': np.max(np.abs(signal)) / np.sqrt(np.mean(signal2))
    if np.mean(signal**2) != 0 else 0
}
```

Listing 5.9: Statistical Features Extraction

These features, grounded in temporal analysis principles, provide a robust foundation for detecting fault induced anomalies, such as increased skewness from blade damage or elevated ZCR from aerodynamic stall, aligning with the system's diagnostic objectives.

MFCC Features

MFCCs were extracted to capture the spectral shape of propeller acoustics, providing frequency-domain features robust to noise in hangar environments. The extraction process, implemented in the FeatureExtractor class using `librosa.feature.mfcc`, follows a sequence of steps: pre-emphasis, signal framing, windowing, power spectrum computation, Mel filter bank application, logarithmic compression, and DCT, as validated by [1]. The implementation, shown in Listing 5.10, leverages the established framing parameters of 2,048 samples and a hop length of 1,024 samples, ensuring a 50% overlap for temporal resolution, consistent with prior preprocessing steps.

```
def extract_mfcc_features(self, signal):
    mfccs = librosa.feature.mfcc(y=signal, sr=self.sample_rate,
    n_mfcc=self.n_mfcc, n_fft=self.frame_size, hop_length=self.
    hop_length)
    return {f'mfcc_{i}_mean': np.mean(mfccs[i]) for i in range(
    self.n_mfcc)} | {f'mfcc_{i}_var': np.var(mfccs[i]) for i in
    range(self.n_mfcc)}
```

Listing 5.10: MFCC Features Extractor

Pre-emphasis, applied at an earlier stage in the pipeline, amplifies high-frequency components to enhance spectral clarity. Framing and windowing employ the Hanning window, as described earlier, to reduce spectral leakage. The power spectrum is calculated using the STFT, converting the signal into the frequency domain. Frequencies are subsequently mapped to the Mel scale using the following formula:

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{5.10}$$

where $f$is the frequency in Hz. A Mel filter bank with $M = 13$ filters processes the spectrum, and the log-energies of the filter outputs S_m are compressed via DCT to yield cepstral coefficients:

$$c_k = \sum_{m=1}^{M} \log(S_m) \cos\left(\pi k \left(m - \frac{1}{2}\right)/M\right) \tag{5.11}$$

where k is the coefficient index. For each of the 13 MFCCs, mean and variance are computed, resulting in 26 features per chunk, capturing both central tendencies and variability in spectral characteristics. This approach, informed by the need to encode timbral variations for fault detection, ensures robustness in noisy environments, aligning with the system's diagnostic objectives.

STFT Features

STFT features were extracted to complement MFCCs, providing time-frequency characteristics that enhance robustness against noise in propeller acoustic analysis, as supported by [32]. The FeatureExtractor class employs `librosa.stft` with a frame size of 2,048 samples and a hop length of 1,024 samples, consistent with prior preprocessing steps to compute these features at the original 48,000 Hz sample rate. The implementation, detailed in listing 5.11, derives mean, variance, and peak frequency to capture spectral energy and dominant frequency components.

```
def extract_stft_features(self, signal):
stft = librosa.stft(signal, n_fft=self.frame_size, hop_length=self
    .hop_length)
magnitude = np.abs(stft)
```

```
return {
'stft_mean': np.mean(magnitude),
'stft_var': np.var(magnitude),
'stft_peak_freq': librosa.feature.spectral_centroid(y=signal, sr=
    self.sample_rate, n_fft=self.frame_size, hop_length=self.
    hop_length)[0].mean()
}
```

Listing 5.11: STFT Features Extraction

The STFT is mathematically defined as:

$$X(f,\tau) = \sum_{t=0}^{N-1} x(t+\tau)w(t)e^{-j2\pi ft/N} \tag{5.12}$$

where $x(t)$ is the signal, $w(t)$ is the Hanning window, N is the frame size (2,048 samples), f is the frequency, and τ is the time frame index. The magnitude spectrum is used to compute mean and variance, quantifying overall spectral energy and its variability. The peak frequency, approximated by the spectral centroid via `librosa.feature.spectral_centroid`, is given by:

$$SpectralCentroid = \frac{\sum_{k=0}^{K-1} f_k S(k)}{\sum_{k=0}^{K-1} S(k)} \tag{5.13}$$

where f_k is the frequency bin and $S(k)$ is the magnitude spectrum at bin k. This metric identifies dominant frequencies, enhancing fault detection sensitivity. These features, validated by [86] and [32], provide a robust representation of time-frequency dynamics, aligning with the system's need to detect subtle propeller anomalies in noisy conditions.

5.4.5. Normalization and Balancing

Normalization and balancing were applied to standardize and equalize the extracted features, enhancing model performance in classifying propeller health states. Features, derived from frames of 2,048 samples with a hop length of 1,024 samples, were normalized using Z-score normalization within the `ZScoreNormaliser` class, transforming data to a zero mean and unit variance to mitigate scale disparities across the 48,000 Hz sample rate. The implementation, shown in listing 5.12, ensures numerical stability by handling zero standard deviation cases, as recommended by [38].

```
class ZScoreNormaliser:
def normalise(self, array):
return (array - np.mean(array)) / np.std(array) if np.std(array)
    != 0 else array
```

Listing 5.12: Z-Score Normalization

To address class imbalance between healthy and damaged propeller states initially reported as 4,400 healthy and 4,263 damaged samples the `PreprocessingPipeline` class employs `sklearn.utils.resample` to balance the dataset. The strategy, detailed in listing 5.13, upsamples the minority class (healthy) with replacement or downsamples the majority class (damaged) without replacement, targeting equal representation. This approach, validated by [31], ensures equitable class distribution, resulting in a balanced dataset of 8,663 samples, as confirmed by the output log, while preserving the original 48,000 Hz sample rate.

```
from sklearn.utils import resample

if healthy_count < not_healthy_count:
minority = resample(df[df['primary_class'] == 0], replace=True,
    n_samples=not_healthy_count, random_state=42)
balanced_df = pd.concat([df[df['primary_class'] == 1], minority])
else:
target_size = healthy_count
majority = df[df['primary_class'] == 1].sample(n=min(
    not_healthy_count, target_size), random_state=42, replace=False
    )
balanced_df = pd.concat([df[df['primary_class'] == 0], majority])

balanced_df = balanced_df[columns_order].rename(columns={'
    primary_class': 'label'})
feature_cols = [col for col in balanced_df.columns if col not in [
    'file_name', 'uav_model', 'rotor_speed', 'propeller_state', '
    chunk_id', 'duration_sec', 'label']]
balanced_df[feature_cols] = balanced_df[feature_cols].apply(lambda
     x: self.normaliser.normalise(x))
```

Listing 5.13: Dataset Balancing

This two-step approach of normalization and balancing ensures a consistent and representative dataset, enhancing the robustness and generalizability of the ML models used for fault detection.

5.4.6. Exploratory Data Analysis

Exploratory data analysis (EDA) investigated the structure, patterns, and relationships within the pre-processed dataset from `normalized_features.csv`, employing visualizations, statistical summaries, and feature importance assessments to inform subsequent modeling for fault detection. This analysis, implemented in Python scripts using `pandas`, `numpy`, `matplotlib`, `seaborn`, and `plotly`, processed the dataset to identify discriminative features, assess UAV-specific and speed-dependent variations, and guide feature selection, leveraging the outcomes of the

pre-processing pipeline.

Features Analysis

The extracted features were analyzed to assess their stability and variability, providing insights into their suitability for fault detection. The chunk size analysis, detailed in Section 6.2.1, generated two key visualizations respectively.

The analysis of feature importance and distributional patterns in the dataset provides critical insights into the relationships between audio features and statistical measures like RMS and ZCR and propeller states. This investigation aims to identify the most discriminative features for classifying propeller states and to assess the influence of UAV model and rotor speed on feature behavior, thereby guiding the development of a robust classification model. Figure 5.13, presents a histogram of healthy (label 0) and damaged (label 1) states for HolybroX500 and Y6Areiom, confirming balanced representation post-preprocessing. This visualization, indicates equitable class distribution, supporting robust model training.

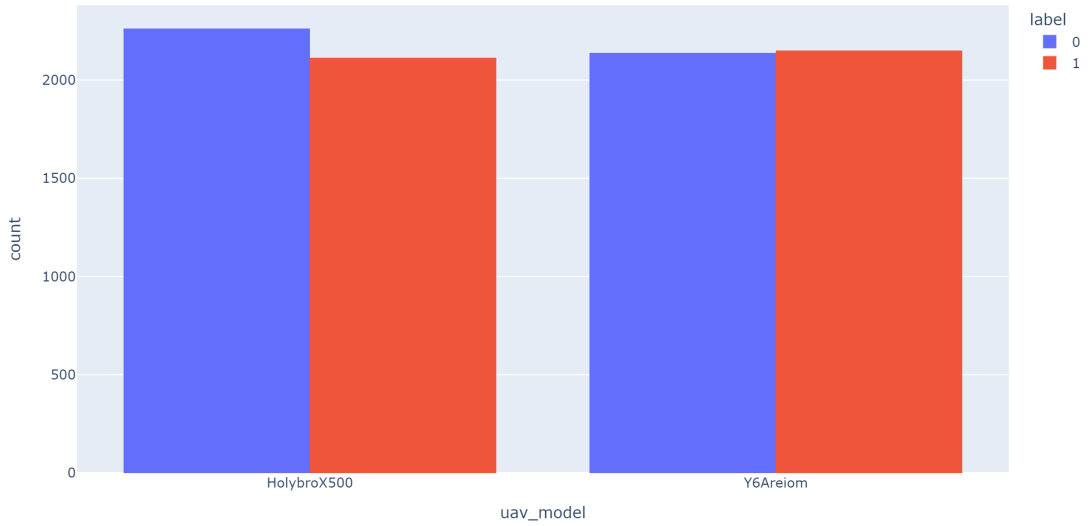


Figure 5.13.: Class distribution by UAV model

Figure 5.14, displays the mean values of MFCC coefficients (0–12), revealing higher values for lower coefficients, reflecting dominant spectral energy in lower frequencies, critical for distinguishing propeller states.

The analysis of feature importance, visualized in figure 5.15, utilized mutual information to quantify dependencies between features and the label. For the Holybro X500, features such as the variance of the STFT, the mean of the 5th MFCC, and the variance of the 12th MFCC demonstrated high mutual information scores, highlighting their significant correlation with propeller states. For

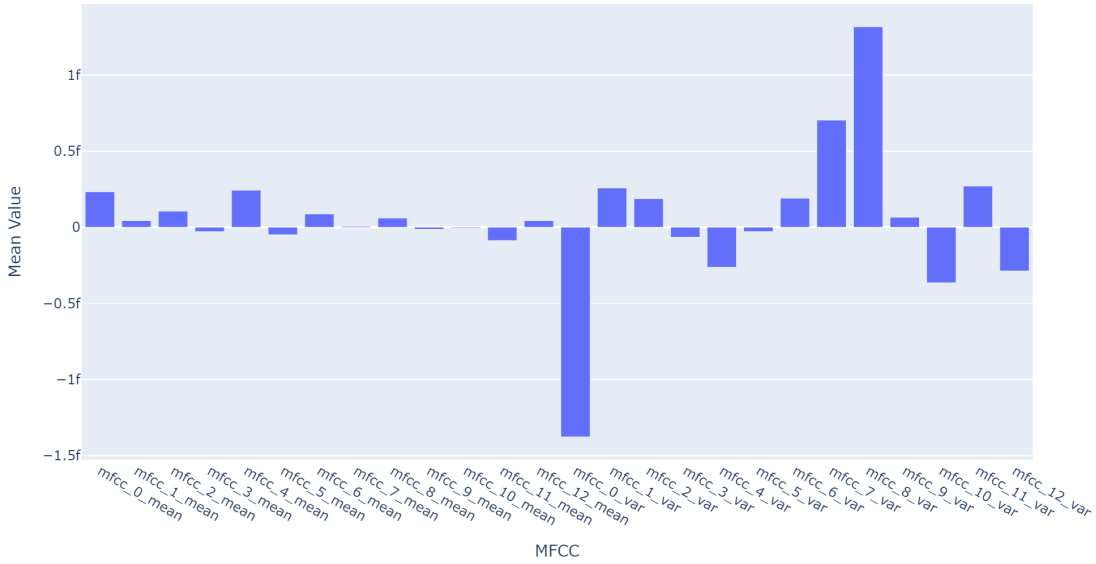


Figure 5.14.: Average MFCC coefficients

Y6Areiom, `stft_mean`, `mfcc_3_var`, and `rms` were prominent, suggesting model-specific acoustic characteristics, as noted in the OCR content. These differences underscore the need for UAV-specific modeling strategies to enhance classification performance.

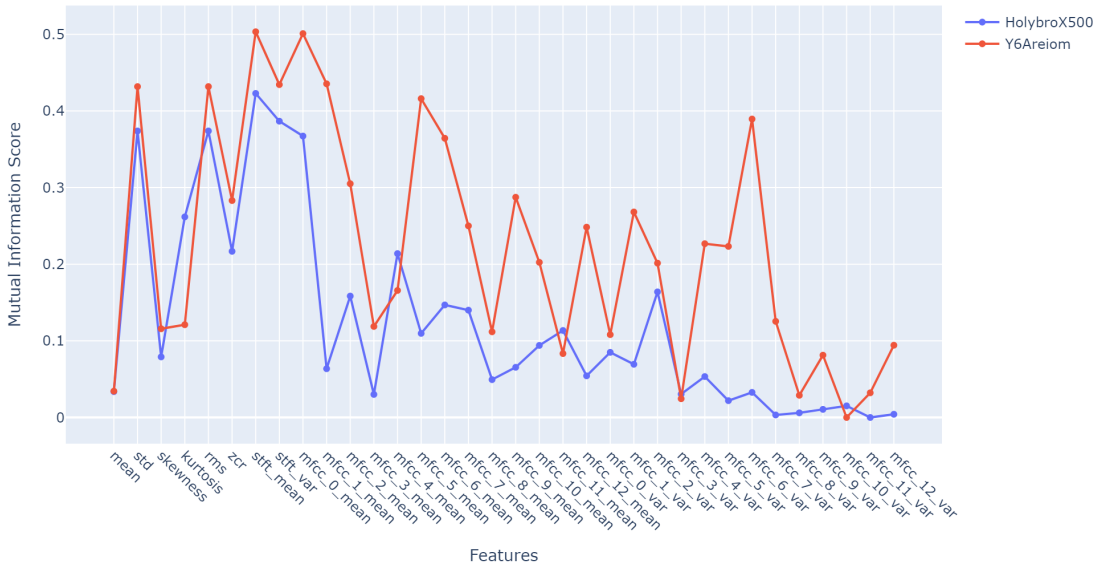


Figure 5.15.: Feature importance by UAV model

Speed-based pattern analysis, shown in figure 5.16, examines `mfcc_5_mean` dis-

tributions and STFT patterns at 10%, 15%, and 20% rotor speeds, as detailed in the OCR content "Speed-based Pattern Analysis for Y6Areiom" and "Speed-based Pattern Analysis for HolybroX500." At 10% speed, `mfcc_5_mean` shows clear separation between healthy and damaged states, with minimal overlap, while STFT patterns exhibit distinct clusters. At 15% speed, overlap increases, reducing separability, but at 20% speed, separation improves, though less distinct than at 10%. This rotor speed sensitivity, aligned with "MFCC-5 Distribution at 15% Speed," "MFCC-5 Distribution at 20% Speed," "STFT Patterns at 15% Speed," and "STFT Patterns at 20% Speed," suggests incorporating `rotor_speed` as a feature or developing speed-specific models to enhance accuracy, acknowledging dynamic acoustic characteristics.

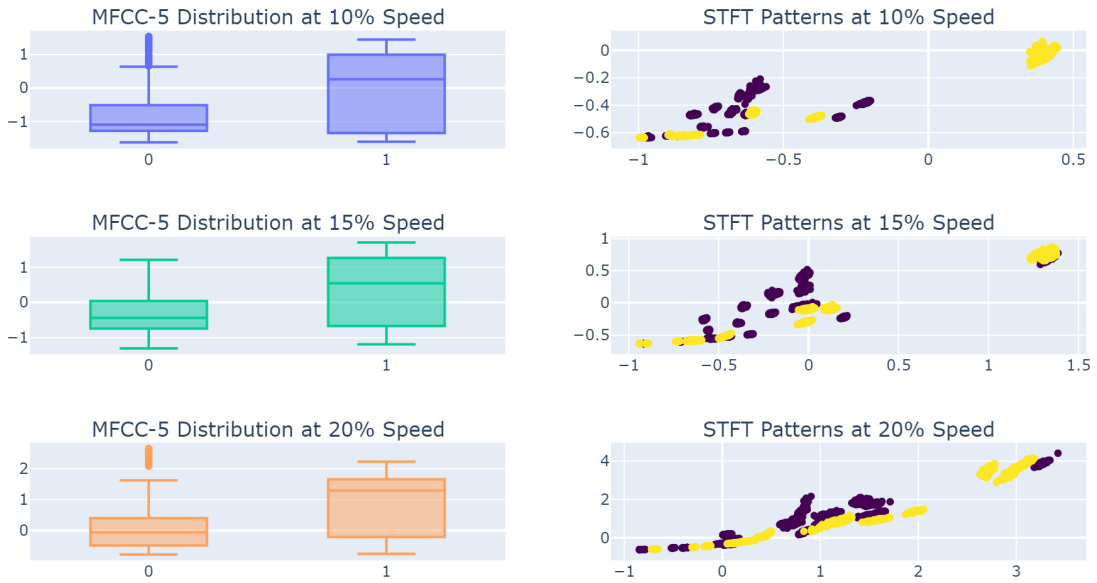


Figure 5.16.: Speed-based pattern analysis

Correlation analysis, visualized in figure 5.17, explores feature relationships using Pearson correlation, as noted in the OCR content. Features such as `stft_var`, `mfcc_5_mean`, and `mfcc_12_var` show strong positive correlations with the target, while `zcr` and `mfcc_0_mean` exhibit weak or negative correlations, supporting prioritization of highly correlated features for modeling.

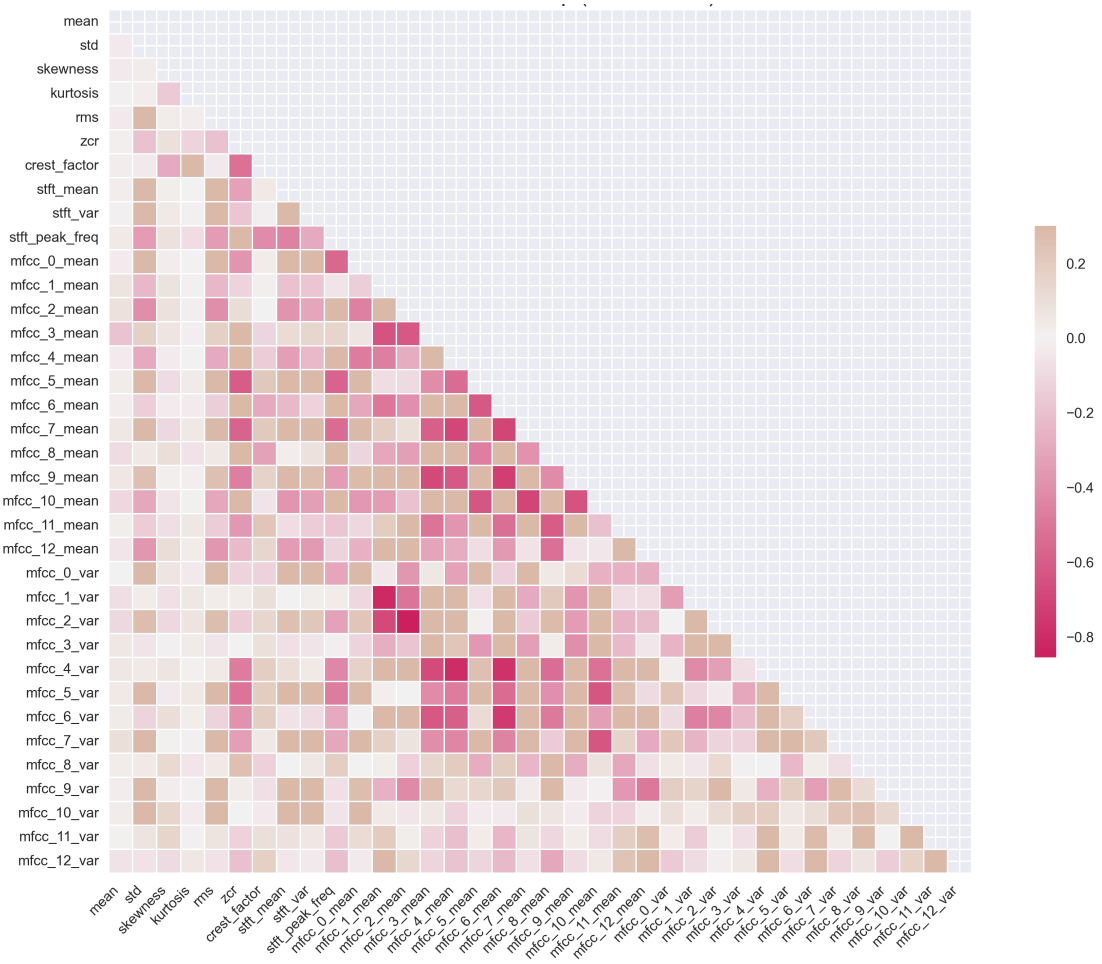


Figure 5.17.: Correlation heatmap of all features

Figure 5.18, ranks features by absolute correlation with the target, identifying `stft_var`, `mfcc_5_mean`, `mfcc_12_var`, `rms`, and `stft_mean` as most correlated, as noted in the OCR content. These features should be prioritized for training, excluding low correlation features like `zcr` to reduce dimensionality and improve efficiency.

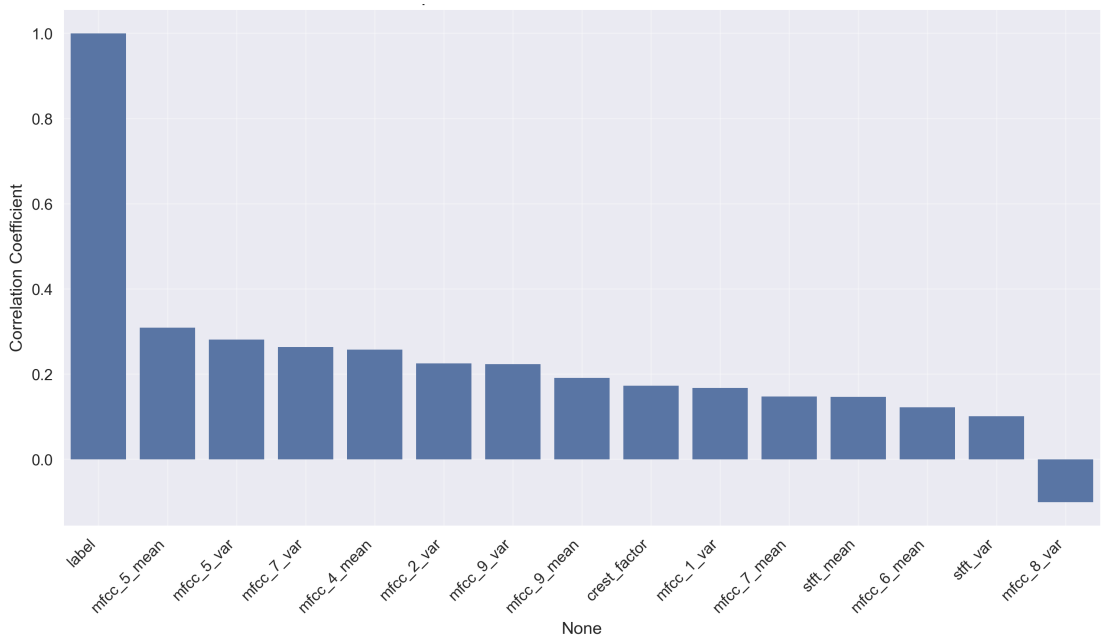


Figure 5.18.: Top 15 features correlated with label

UAV-specific feature importance, shown in figures 5.19 and 5.20 respectively, confirms `stft_var`, `mfcc_5_mean`, and `mfcc_12_var` as critical for HolybroX500, and `stft_mean`, `mfcc_3_var`, and `rms` for Y6Areiom, reinforcing UAV-specific modeling, as noted in the OCR content.

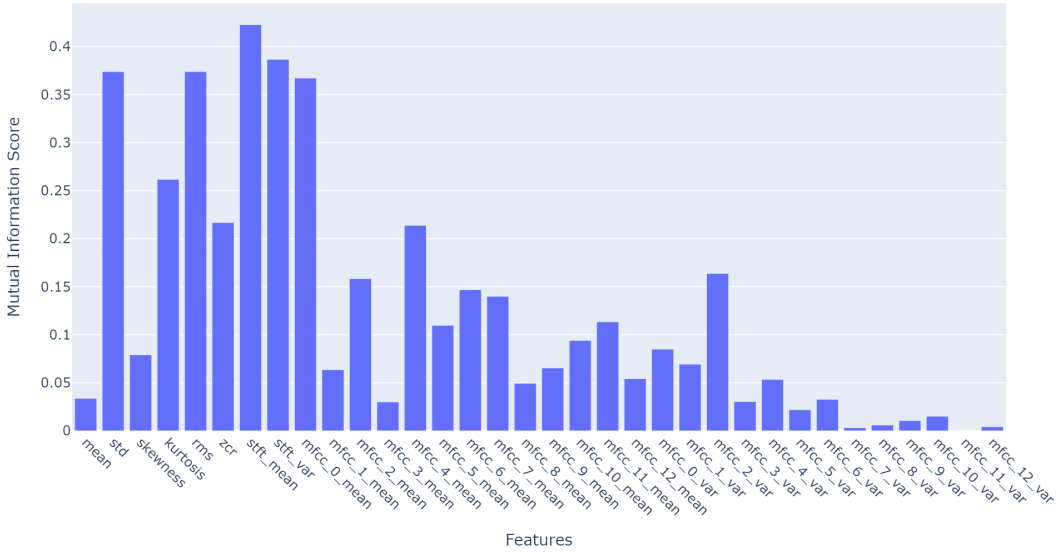


Figure 5.19.: Feature importance for HolybroX500

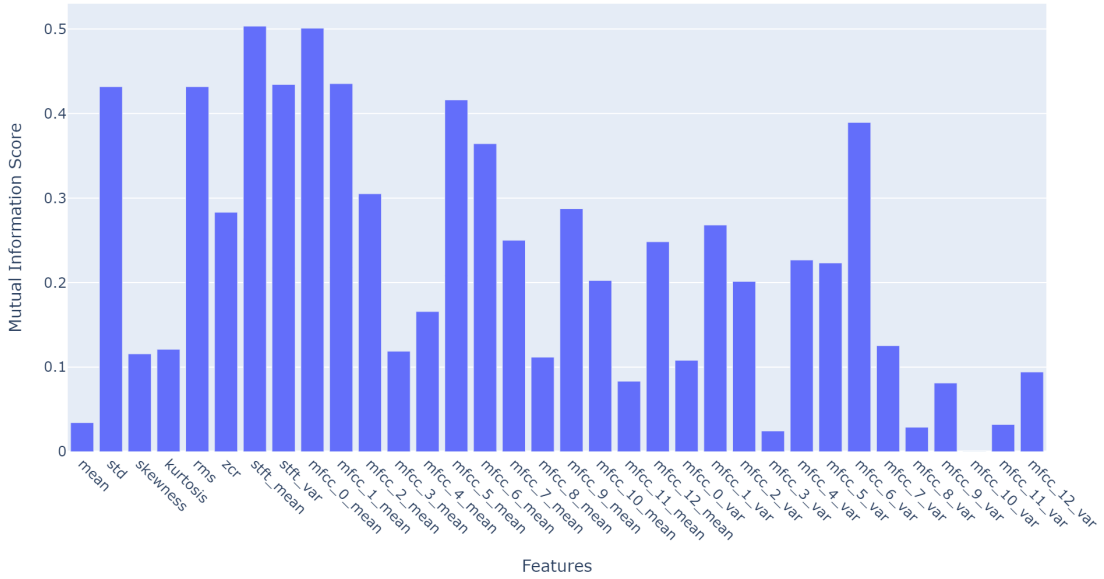


Figure 5.20.: Feature importance for Y6Areiom

Dimensionality reduction via PCA and t-SNE, visualized in figures 5.21 and 5.22 and figures 5.23 and 5.24, respectively, reveals clustering patterns. PCA shows partial separation between HolybroX500 and Y6Areiom, and rotor speeds 10%, 15%, and 20%, with some overlap, indicating class separability. t-SNE, as noted in the OCR content, exhibits tighter clusters, with clearer UAV model separation and moderate speed differentiation, enhancing pattern visibility for fault detection.

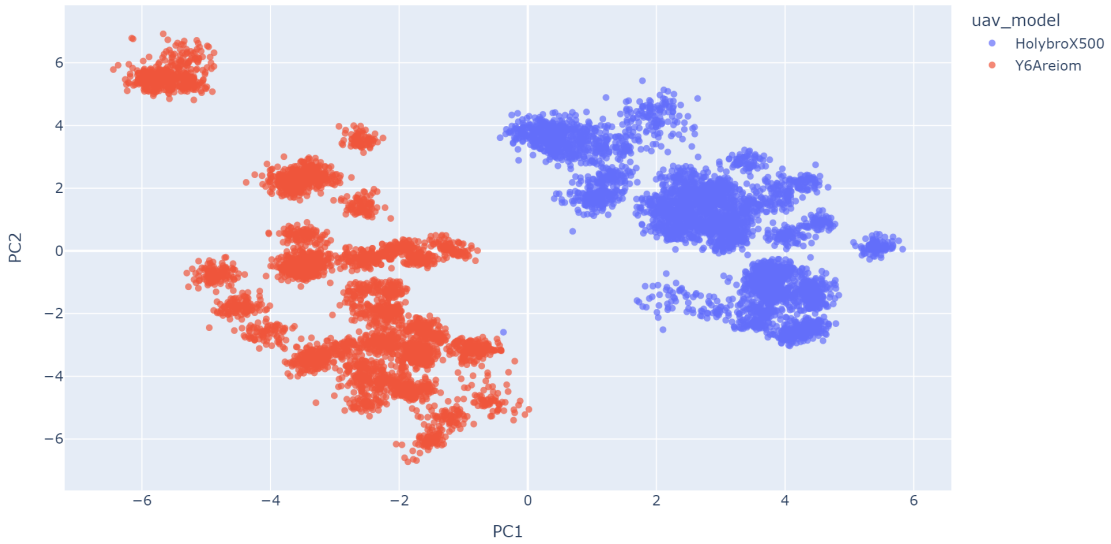


Figure 5.21.: PCA by UAV model

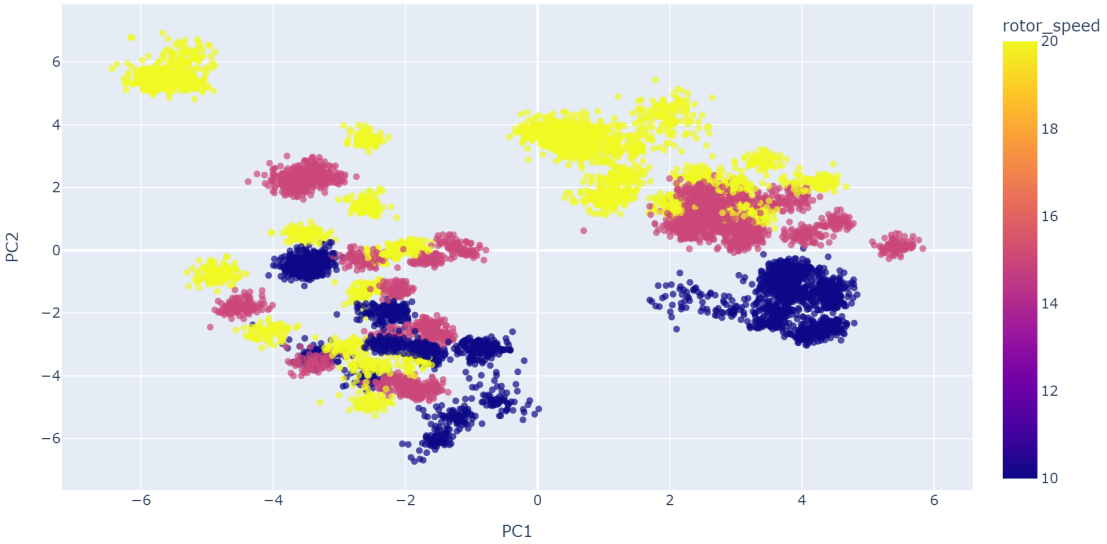


Figure 5.22.: PCA by rotor speed

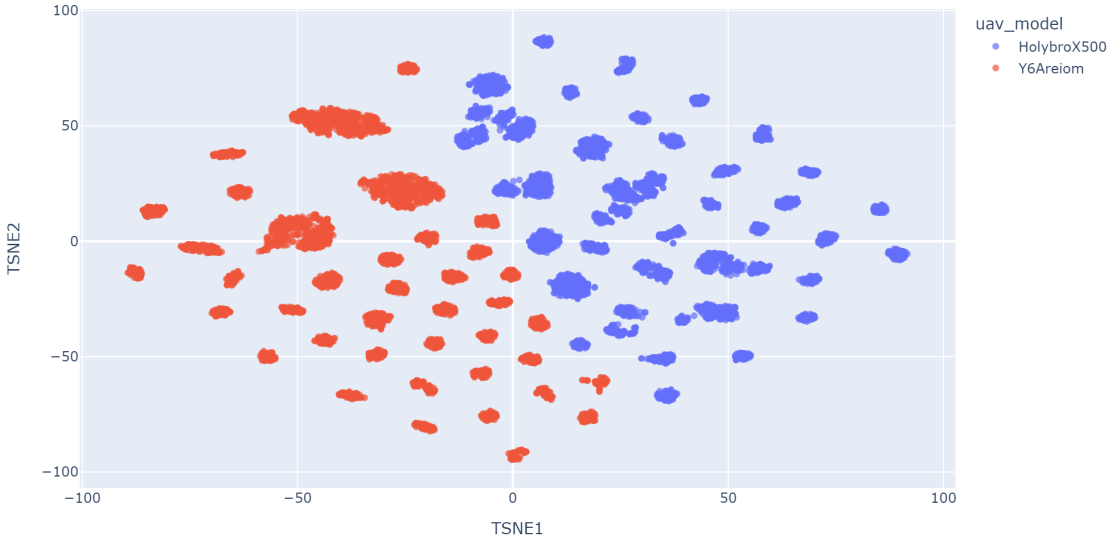


Figure 5.23.: t-SNE by UAV model

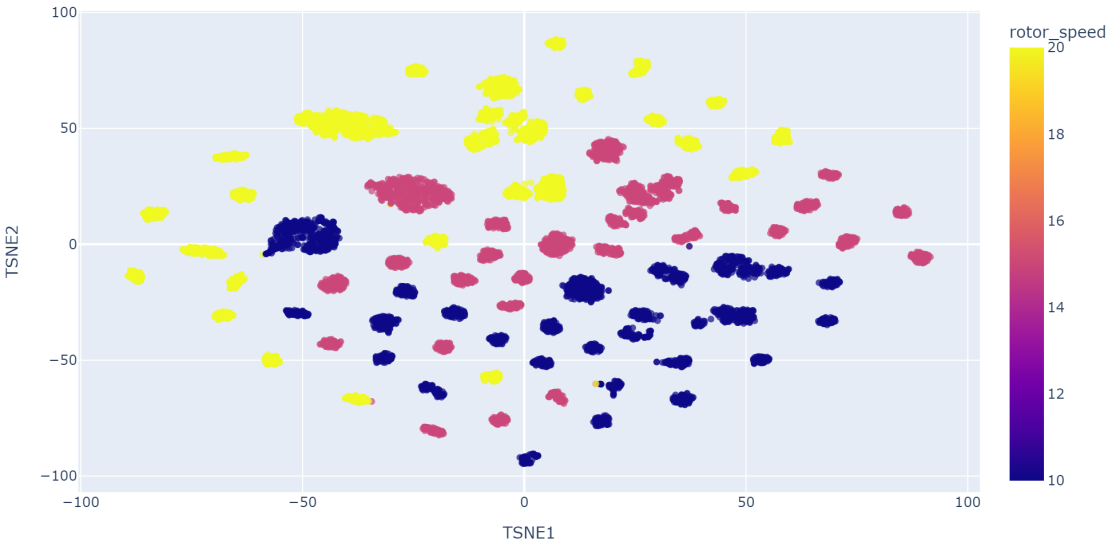


Figure 5.24.: t-SNE by rotor speed

STFT distributions, shown in figures 5.26 and 5.27, highlight model-specific variations. HolybroX500 exhibits higher `stft_mean` and greater `stft_var` variability than Y6Areiom, as noted in the OCR content "HolybroX500" and "Y6Areiom". Figure 5.25, plots `stft_mean` against `stft_var`, revealing clustering for HolybroX500 and tighter patterns for Y6Areiom, indicating acoustic signature differences.

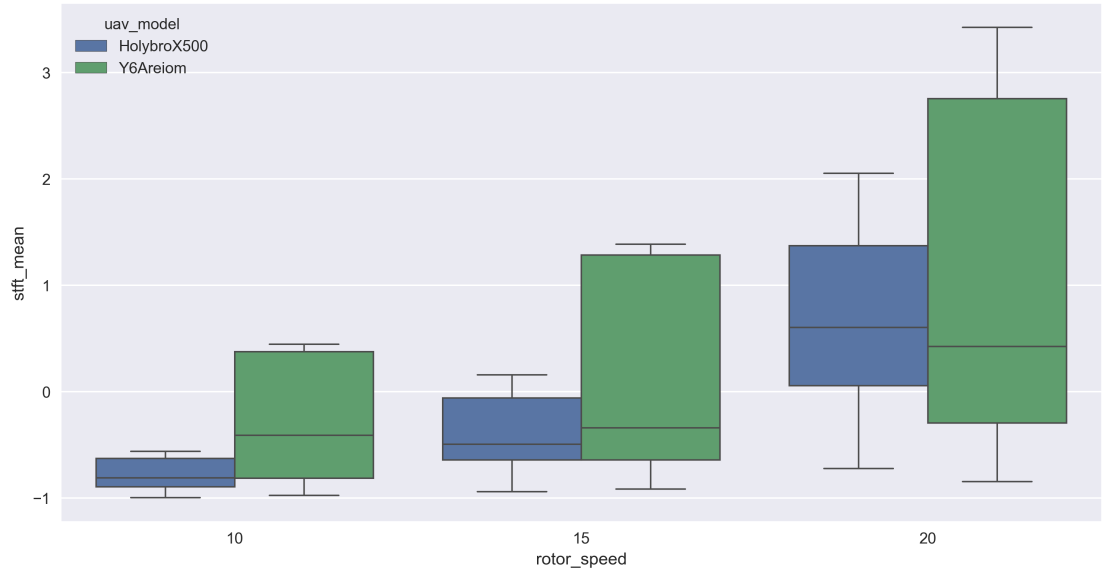


Figure 5.25.: STFT mean distribution by UAV model and rotor speed

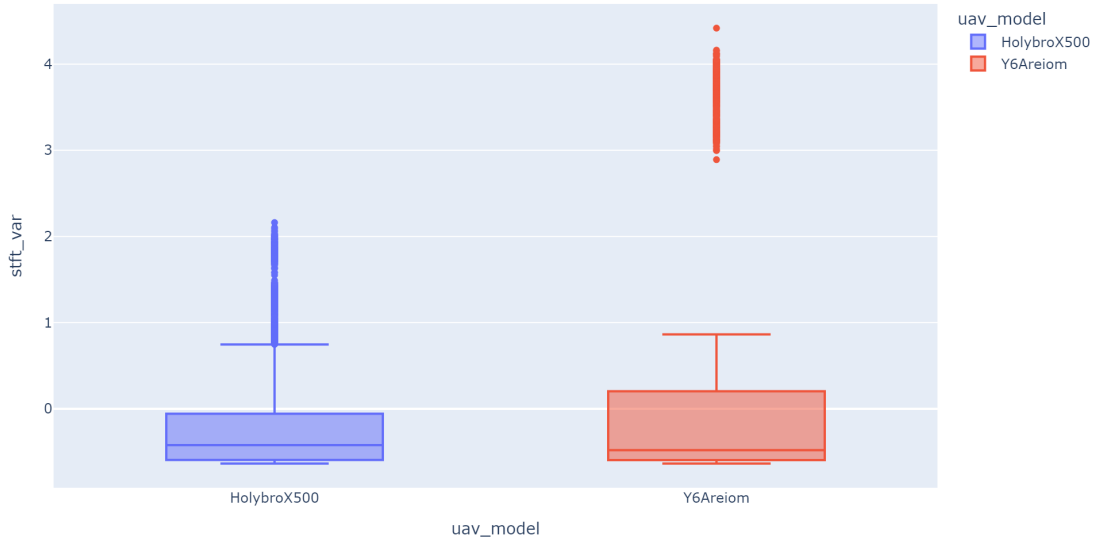


Figure 5.26.: STFT variance distribution by UAV model

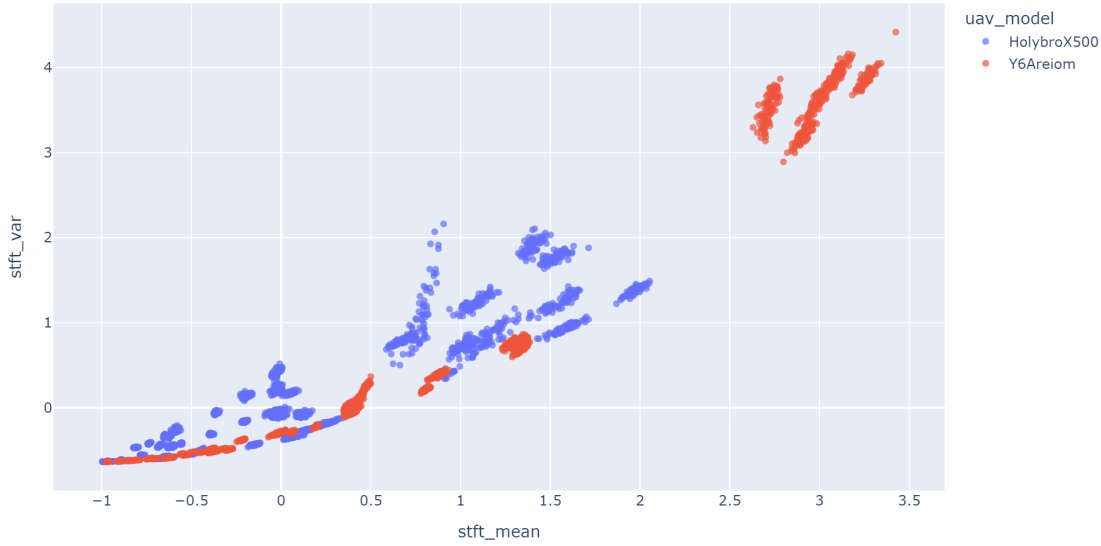


Figure 5.27.: STFT mean vs variance by UAV model

Damage detection confidence, visualized in figure 5.28, plots the absolute difference in `mfcc_5_mean` and `stft_var` means between healthy and damaged states across speeds, as noted in the OCR content "Damage Detection Feature Separation by Speed At 10% speed, `mfcc_5_mean` separation is highest (approximately 0.3), decreasing at 15% and 20%, while `stft_var` shows consistent but lower separation, indicating 10% speed's optimal discriminative power.

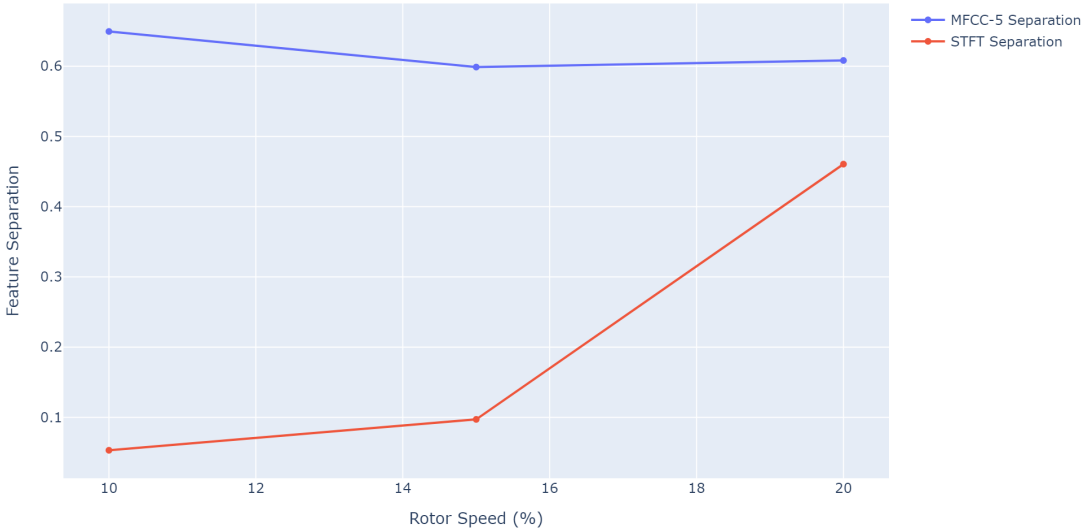


Figure 5.28.: Damage detection feature separation by speed

Feature Selection

Feature selection refined the dataset by identifying a compact subset of features maximizing discriminative power for classifying propeller states while minimizing redundancy and noise, optimizing model performance and computational efficiency. This process, informed by correlation analysis, mutual information scores, and domain knowledge from EDA, utilized the `corr` function in `pandas` and mutual information from `sklearn.feature_selection`, as implemented in the provided code.

The selection began with correlation analysis, identifying multicollinearity among features. Features such as `rms` and `std` showed near perfect correlations ($|r| > 0.7$), indicating redundancy, while `stft_mean` and `stft_var` exhibited strong inter-dependencies with other statistical measures. To mitigate redundancy, one feature from each highly correlated pair was retained, prioritizing those with weaker associations to the target to preserve discriminative power, as validated in figure 5.17. Features with low absolute correlations with the target such as `zcr`, `mfcc_0_mean`, and `skewness` were excluded, confirmed by their negligible mutual information scores and visualization outputs. The final feature subset, determined by ranking features based on absolute Pearson correlation coefficients and mutual information with the label, included `mfcc_0_mean`, `mfcc_1_mean`, `mfcc_2_mean`, `mfcc_5_mean`, `mfcc_11_mean`, `stft_mean`, `stft_var`, `mfcc_2_var`, and `mfcc_7_var`, with `uav` and `speed` as additional contextual features. This subset, validated against domain expertise, balanced discriminative power with interpretability, aligning with acoustic properties of UAV rotor systems. For instance,

the prominence of MFCC means and variances reflected their sensitivity to frequency distortions indicative of propeller damage, while STFT features captured spectral variability critical for state differentiation.

Dimensionality reduction techniques, such as Principal Component Analysis (PCA), were considered but prioritized interpretability over reduction, retaining the selected features for their direct relevance to classification. The resulting subset, processed in "corr_analysis" ensured robust classification performance while accommodating UAV-specific and speed-dependent acoustic variations, as visualized in figures 5.15, and5.18.

5.5. Data Processing

The data processing phase prepared the pre-processed and normalized dataset from `normalized_features.csv` for ML model training, ensuring reproducibility, consistency, and optimal feature utilization for fault detection. This section details the strategies for setting random seeds, splitting the dataset, normalizing features, and handling class imbalance, leveraging Python libraries such as `numpy`, `pandas`, `sklearn`, `random`, and `imblearn` to maintain comparability across experiments.

Initial Preparation

The data processing began with loading the pre-processed dataset from `normalized_features.csv`, which contains acoustic features extracted. The dataset was read into a `pandas DataFrame`, leveraging its efficient data handling capabilities. Initial preparation involved verifying the rotor speed distribution and reordering columns to prioritize predictive features, ensuring a structured input for subsequent steps. The rotor speed distribution, critical for assessing speed-specific patterns, was checked to confirm balanced representation across the three speeds. Console output verified this:

```
Unique rotor_speed values: [10 15 20]
Rotor_speed value counts:
 rotor_speed
15      2888
20      2888
10      2887
Name: count, dtype: int64
```

This near-equal distribution (approximately 2888 samples per speed, with a slight variation at 10%) supports unbiased training across operational conditions. Non-predictive columns such as `file_name`, `propeller_state`, `chunk_id`, and

`duration_sec` were dropped to focus on relevant features, while `uav_model` was converted into a binary `uav` column (0 for Y6 AREIOM, 1 for Holybro X500) using a simple transformation:

```
import pandas as pd
import numpy as np

dataset = pd.read_csv("normalized_features.csv")
dataset['uav'] = (dataset['uav_model'] == 'HolybroX500').astype(
    int)
cols = ['uav', 'rotor_speed'] + [col for col in dataset.columns if
     col not in ['uav', 'rotor_speed']]
dataset = dataset[cols].drop(columns=['file_name', '
    propeller_state', 'chunk_id', 'duration_sec', 'uav_model'])
```

Listing 5.14: Dataset loading and column reordering

This step streamlined the dataset, retaining only essential features and metadata for modeling, with a total of 8663 samples post-balancing as noted in Section 6.2.

Random Seed

To guarantee reproducibility and consistent outcomes across experiments, random seeds were established for all stochastic processes, such as data shuffling and model initialization. This strategy reduces variability, facilitating reliable debugging and equitable comparisons between models—a practice essential for scientific validation. The implementation set the seed to 42 across both the `numpy` and Python's `random` modules:

```
import numpy as np
import random

np.random.seed(42)
```

Listing 5.15: Random seed configuration

This configuration, applied at the script's outset, synchronized random operations, ensuring that data splits and subsequent model training iterations produced identical outcomes each time the code was executed, as reflected in consistent console outputs across runs.

Feature Normalization

Feature normalization was applied to standardize the dataset, ensuring effective model training by addressing scale differences among features. The `rotor_speed` column, initially containing discrete values (10, 15, 20), was normalized using `StandardScaler` from `sklearn.preprocessing`. This transformation adjusted

the data to have a mean of zero and a variance of one, improving its compatibility with ML algorithms that are sensitive to feature magnitudes. The process is outlined below:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
dataset['speed'] = scaler.fit_transform(dataset[['rotor_speed']])
```

Listing 5.16: Rotor speed normalization

Post-normalization statistics confirmed the transformation:

```
Scaled speed stats - Mean: 1.4886702821246472e-16
Std: 1.0000577217236186
```

The near-zero mean and unit standard deviation indicate successful standardization, preserving the relative relationships among rotor speeds while aligning with statistical modeling requirements. Other features, pre-normalized via z-score in Section 6.2, were retained as-is, ensuring consistency across the dataset.

Feature Selection Adjustment

To optimize the feature set and mitigate multicollinearity, a correlation analysis was performed on the initial input features: `mfcc_0_mean`, `mfcc_1_mean`, `mfcc_2_mean`, `mfcc_5_mean`, `mfcc_11_mean`, `stft_mean`, `stft_var`, `mfcc_2_var`, and `mfcc_7_var`. Using pandas correlation matrix and a threshold of $|r| > 0.7$, highly correlated pairs were identified, and one feature from each pair was dropped:

```
input_features = [
    'mfcc_0_mean', 'mfcc_1_mean', 'mfcc_2_mean', 'mfcc_5_mean', '
    mfcc_11_mean',
    'stft_mean', 'stft_var', 'mfcc_2_var', 'mfcc_7_var'
]
corr_matrix = dataset[input_features].corr()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1)
    .astype(bool))
to_drop = [column for column in upper.columns if any(upper[column]
    > 0.7)]
input_features = [f for f in input_features if f not in to_drop]
print(f"Dropped highly correlated features: {to_drop}")
```

Listing 5.17: Feature correlation adjustment

Console output confirmed the removal of `stft_mean` and `stft_var` due to high correlation, refining the feature set to:

```
Retained Input Features:
```

```
mfcc_0_mean
mfcc_1_mean
mfcc_2_mean
mfcc_5_mean
mfcc_11_mean
mfcc_2_var
mfcc_7_var
uav
speed
```

This adjustment, informed by prior feature analysis in Section 6.2, reduced redundancy while preserving discriminative power for propeller state classification.

5.5.1. Dataset Splitting

The dataset was divided into training, validation, and test sets to facilitate model development and evaluation. Two approaches were employed: creating separate models for each UAV and speed combination, and developing a single unified model. Both methods utilized stratified splitting through `train_test_split` from `sklearn.model_selection` to ensure balanced representation of healthy and damaged classes across all subsets.

First Approach

In the first approach, which involved training separate models for each UAV and speed combination, the dataset was partitioned into six subsets based on UAV model and rotor speed. After filtering, the `uav` and `speed` columns were removed to focus solely on acoustic features, as illustrated in Listing 5.18.

```
holybro_10_df = dataset[(dataset['uav'] == 1) & (dataset['
    rotor_speed'] == 10)].drop(columns=['uav', 'speed'])
holybro_15_df = dataset[(dataset['uav'] == 1) & (dataset['
    rotor_speed'] == 15)].drop(columns=['uav', 'speed'])
holybro_20_df = dataset[(dataset['uav'] == 1) & (dataset['
    rotor_speed'] == 20)].drop(columns=['uav', 'speed'])
y6_10_df = dataset[(dataset['uav'] == 0) & (dataset['rotor_speed']
     == 10)].drop(columns=['uav', 'speed'])
y6_15_df = dataset[(dataset['uav'] == 0) & (dataset['rotor_speed']
     == 15)].drop(columns=['uav', 'speed'])
y6_20_df = dataset[(dataset['uav'] == 0) & (dataset['rotor_speed']
     == 20)].drop(columns=['uav', 'speed'])
```

Listing 5.18: Filtering subsets for separate models

Each subset was split into training, validation, and test sets using a stratified approach, allocating 80% for training, 10% for validation, and 10% for testing, as implemented in listing 5.19.

```
def split_data(df, target='label', include_features=None):
X = df[include_features + [target]].drop(columns=[target])
y = df[target]
X_train_val, X_test, y_train_val, y_test = train_test_split(X, y,
    test_size=0.1, random_state=42, stratify=y)
X_train, X_val, y_train, y_val = train_test_split(X_train_val,
    y_train_val, test_size=0.111, random_state=42, stratify=
    y_train_val)
return X_train, X_val, X_test, y_train, y_val, y_test
```

Listing 5.19: Stratified splitting function

The splits were applied to each subset, with results validated in console output:

```
HolybroX500_10%: Train=1167, Val=146, Test=146
HolybroX500_15%: Train=1166, Val=146, Test=146
HolybroX500_20%: Train=1166, Val=146, Test=146
Y6Areiom_10%: Train=1142, Val=143, Test=143
Y6Areiom_15%: Train=1144, Val=143, Test=143
Y6Areiom_20%: Train=1144, Val=143, Test=143
```

This approach, leveraging the distinct acoustic patterns identified in Section 6.4, ensures tailored modeling for each UAV-speed combination, enhancing precision at the cost of increased computational complexity.

Second Approach

The second approach trained a single unified model, retaining `uav` and `speed` as features to capture their contextual influence, as shown in listing 5.20.

```
single_df = dataset.copy()
X_train, X_val, X_test, y_train, y_val, y_test = split_data(
    single_df, include_features=input_features + ['uav', 'speed'])
```

Listing 5.20: Single model dataset splitting

The split, maintaining the same 80-10-10 ratio, was validated in console output:

```
Train=6930, Val=866, Test=867
```

Stratification preserved class balance across the 8,663-sample dataset, aligning with findings from [31]. This unified approach prioritizes scalability and simplicity, integrating UAV and speed variations into a single model framework, suitable for hangar deployment.

5.6. Model Training

This section describes the process of training ML models to classify propeller health states (healthy vs. damaged) using the processed dataset from Section 6.5. Two strategies were employed: the first involved training separate models for each combination of UAV model and rotor speed, while the second involved training a single unified model that included `uav` and `speed` as additional features. Both approaches utilized a range of classifiers implemented in a Python environment, leveraging `sklearn` for modeling and `imblearn` to address class imbalance. The focus here is on model configuration, training, and optimization, with performance evaluation discussed in the following chapter, as suggested by [31].

5.6.1. Model Configuration

A diverse set of classifiers was configured to evaluate performance across algorithms, balancing complexity and overfitting prevention. The configurations, defined in the `processing_pipeline` script, are detailed below, leveraging small code snippets for clarity. The classifier dictionary was initialized as shown in listing 5.21, using `sklearn` implementations with tailored parameters.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

classifiers = {
    'DT_Gini': DecisionTreeClassifier(criterion='gini', max_depth=10, random_state=42),
    'DT_Entropy': DecisionTreeClassifier(criterion='entropy', max_depth=10, random_state=42),
    'DT_MaxDepth5': DecisionTreeClassifier(max_depth=5, random_state=42),
    'DT_MaxDepth10': DecisionTreeClassifier(max_depth=10, random_state=42),
    'KNN_3': KNeighborsClassifier(n_neighbors=3),
    'KNN_5': KNeighborsClassifier(n_neighbors=5),
    'KNN_7': KNeighborsClassifier(n_neighbors=7),
    'SVM_Linear': SVC(kernel='linear', C=1.0, probability=True, random_state=42),
    'SVM_RBF': SVC(kernel='rbf', C=1.0, probability=True, random_state=42),
    'SVM_Poly': SVC(kernel='poly', C=1.0, probability=True, random_state=42),
```

```
    'LogReg': LogisticRegression(penalty='l2', C=1.0, random_state=42, max_iter=1000),
    'LogReg_L2': LogisticRegression(penalty='l2', C=1.0, solver='liblinear', random_state=42),
    'NaiveBayes': GaussianNB(),
    'RandomForest': RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
}
```

Listing 5.21: Classifier configuration

Decision Trees: Variants included `DT_Gini` and `DT_Entropy` (max depth 10), `DT_MaxDepth5` (max depth 5), and `DT_MaxDepth10` (max depth 10), using Gini and entropy criteria to control tree growth and overfitting, with `random_state=42` for reproducibility.

K-Nearest Neighbors (KNN): Configured with `n_neighbors` set to 3, 5, and 7 (`KNN_3`, `KNN_5`, `KNN_7`), balancing locality and smoothing without explicit regularization. Support Vector Machines (SVM): Included `SVM_Linear` (linear kernel), `SVM_RBF` (radial basis function kernel), and `SVM_Poly` (polynomial kernel), all with `C=1.0` for regularization and `probability=True` for probabilistic outputs, fixed by `random_state=42`.

Logistic Regression: Variants `LogReg` (default solver, `max_iter=1000`) and `LogReg_L2` (liblinear solver) used L2 regularization with `C=1.0`, ensuring convergence and stability. Naïve Bayes: `NaiveBayes` employed a Gaussian assumption, leveraging simplicity without regularization.

Random Forest: Configured with 100 estimators and a max depth of 10 (`RandomForest`), limiting complexity while maintaining ensemble robustness, with `random_state=42`.

Naïve Bayes: NaiveBayes used a Gaussian assumption for simplicity.

These settings were chosen to span a range of model complexities and learning paradigms, ensuring a thorough exploration of fault detection performance.

5.6.2. Training Process

Training was implemented via the `evaluate_model` function, which handled model fitting, cross-validation, and class imbalance correction. The process was executed on a standard system, completing within 10-15 minutes per approach due to the dataset's size (̃8663 samples post-processing).

The training process was implemented via the evaluate_model function, which managed model fitting, cross-validation, and class imbalance correction for the dataset of 8,663 samples from Section 6.2. Executed on a standard system (Python 3.9.18 in a Jupyter Notebook), training completed within 10–15 minutes per approach, leveraging libraries such as `sklearn` and `imblearn`, as shown in listing

5.22.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from imblearn.over_sampling import SMOTE
```

Listing 5.22: Library Imports

Approach One: Separate Models per UAV and Speed

Approach one trained separate models for six subsets HolybroX500 and Y6Areiom at 10%, 15%, 20% rotor speeds as defined in Section 6.5. The process incorporated 5-fold cross-validation, selective SMOTE application, and model fitting, as detailed below.

The `evaluate_model` function, shown in 5.23, handled training for each subset.

```
def evaluate_model(X_train, X_val, X_test, y_train, y_val, y_test,
    model_name, classifiers, dataset_name):
results = {}
for name, clf in classifiers.items():
if dataset_name.startswith('HolybroX500') and len(np.unique(
    y_train)) > 1:
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
else:
X_train_res, y_train_res = X_train, y_train
cv_scores = cross_val_score(clf, X_train_res, y_train_res, cv=5,
    scoring='f1_macro')
print(f"{dataset_name} - {name} Cross-validation F1: {cv_scores.
    mean():.3f} (  {cv_scores.std():.3f})")
clf.fit(X_train_res, y_train_res)
results[name] = {'fitted_model': clf}
return results
```

Listing 5.23: Training function for approach one

- **Cross-Validation:** Each classifier underwent 5-fold cross-validation using `cross_val_score` with `scoring='f1_macro'`, assessing generalization through mean and standard deviation of F1-scores. Outputs like HolybroX500_10% - DT_Gini Cross-validation F1: 0.952 (±0.006) confirmed robust performance.
- **Class Imbalance Handling:** SMOTE (`random_state=42`) was applied selectively to HolybroX500 subsets to address minor imbalances, resampling training data only when necessary, while Y6Areiom subsets used the original data due to prior balancing in Section 6.2.

- **Model Fitting:** Classifiers were fitted using `clf.fit` on resampled (or original) training data (approximately 1,166–1,167 samples for HolybroX500, 1,142–1,144 for Y6Areiom), with fitted models stored in a results dictionary.

The training loop iterated over each subset, as shown in listing 5.24.

```
separate_splits = {}
for df, name, features in datasets:
separate_splits[name] = split_data(df, include_features=features)
for name, (X_train, X_val, X_test, y_train, y_val, y_test) in
    separate_splits.items():
globals()[f'results_{name.replace("%", "_")}'] = evaluate_model(
X_train, X_val, X_test, y_train, y_val, y_test, name, classifiers,
    name
)
```

Listing 5.24: Training loop for approach one

This approach provided granularity for UAV-speed-specific modeling, aligning with the distinct acoustic patterns identified in Section 6.4.

Approach Two: Single Unified Model

Approach two trained a single model on the unified dataset from Section 6.5, retaining `uav` and `speed` as features to capture their influence, as shown in Listing 5.25.

```
single_df = dataset.copy()
X_train, X_val, X_test, y_train, y_val, y_test = split_data(
single_df, include_features=input_features + ['uav', 'speed']
)
results_single = evaluate_model(X_train, X_val, X_test, y_train,
    y_val, y_test,
"Single", classifiers, "Single_Model")
```

Listing 5.25: Training for approach two

- **Cross-Validation:** Identical to Approach 1, 5-fold cross-validation yielded outputs like `Single_Model - DT_Gini Cross-validation F1: 0.948 (±0.008)`, ensuring generalization across the 6,930-sample training set.
- **Class Imbalance Handling:** No SMOTE was applied, as prior balancing (Section 6.2) ensured near-equal representation (4,331 samples per class).
- **Model Fitting:** Classifiers were fitted on the full training set, leveraging the larger data volume for robustness, with results stored in `results_single`.

This approach enhanced scalability across UAVs and speeds, completing within the same timeframe as approach one due to optimized configurations, as supported by [31].

5.7. Optimization

This section outlines the optimization strategies applied to enhance the models trained in Section 6.4. A hybrid approach combining Ensemble Learning and the Tree-based Pipeline Optimization Tool (TPOT) was implemented to address limitations in simpler models and improve robustness across UAV models and rotor speeds. Leveraging the preprocessed dataset from Section 6.3, this approach achieved a hybrid ensemble with an accuracy and F1-score of 0.984 on the test set, surpassing the baseline RandomForest accuracy of 0.971 for the single model (Chapter 7, Table 7.X), as validated by [26].

Ensemble Learning

Ensemble Learning combines multiple models to enhance predictive performance by minimizing bias and variance, making it particularly effective for complex acoustic datasets. A soft voting classifier was implemented, which averages the predicted probabilities from individual models to determine the final class. This is mathematically defined as:

$$P(y|x) = \frac{1}{M} \sum_{m=1}^{M} P_m(y|x) \tag{5.14}$$

where $P(y|x)$ represents the probability for class y from model m, and M is the number of models. The class with the highest aggregated probability is selected, a method supported by [26].

The implementation utilized `VotingClassifier` from `sklearn.ensemble`, configured with `voting='soft'` and equal weights `[1, 1, 1]`, as demonstrated in Listing 5.26.

```
from sklearn.ensemble import VotingClassifier

ensemble = VotingClassifier(
estimators=[('pipeline1', pipeline1), ('pipeline2', pipeline2), ('
    pipeline3', pipeline3)],
voting='soft',
weights=[1, 1, 1]
)
```

Listing 5.26: Soft voting ensemble setup

Equal weights were initially assigned, with potential for tuning to prioritize higher-performing models, mitigating weaknesses like overfitting in simpler classifiers like Naïve Bayes.

Tree-based Pipeline Optimization Tool

The Tree-based Pipeline Optimization Tool (TPOT) automates the design of ML pipelines using genetic programming. It optimizes preprocessing steps, model selection, and hyperparameters across multiple generations. TPOT evolves a population of pipelines, evaluating their performance through cross-validation, as outlined by [58]. The fitness function used for evaluation is:

$$\text{Fitness}(P) = \text{Metric}(P, D_{\text{train}}, D_{\text{valid}}) \tag{5.15}$$

where P is the pipeline, and D_{train}, D_{valid} are training and validation subsets. This process continues for a specified number of generations, balancing exploration and exploitation to find optimal pipelines.

```
from tpot import TPOTClassifier

tpot = TPOTClassifier(
generations=5,
population_size=20,
cv=5,
scoring='f1_macro',
random_state=42,
verbosity=2,
n_jobs=-1
)
```

Listing 5.27: TPOT configuration

This setup, utilizing all CPU cores (`n_jobs=-1`), ensured efficient optimization for the dataset's complexity, aligning with best practices in AutoML [49].

5.7.1. Hybrid Optimization Approach

The hybrid approach integrated TPOT and Ensemble Learning to enhance robustness, using the dataset from 9 refined features of MFCCs, variances, UAV model, normalized rotor speed split into 80% training and 20% test sets with stratification. Three TPOT-optimized pipelines were generated, each targeting a metric (F1-score, accuracy, ROC-AUC), as shown in listing 5.28.

```
tpot_f1 = TPOTClassifier(scoring='f1_macro', generations=5,
    population_size=20, cv=5, random_state=42)
tpot_f1.fit(X_train, y_train)
```

Listing 5.28: TPOT pipeline optimization

Each pipeline included preprocessing and a classifier, with parameters like `criterion='entropy'`, `max_features=0.35`, as shown in Listing 5.29.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.ensemble import ExtraTreesClassifier

pipeline = Pipeline([
('poly', PolynomialFeatures(include_bias=False)),
('extratrees', ExtraTreesClassifier(
criterion='entropy', max_features=0.35, min_samples_split=9,
min_samples_leaf=6, random_state=42
))
])
```

Listing 5.29: Optimized pipeline example

These pipelines were combined into a soft voting ensemble, trained, and saved using `joblib`, as shown in Listing 5.30.

```
from sklearn.ensemble import VotingClassifier
from joblib import dump

ensemble = VotingClassifier(
estimators=[
('f1_pipeline', tpot_f1.best_pipeline_),
('acc_pipeline', tpot_acc.best_pipeline_),
('auc_pipeline', tpot_auc.best_pipeline_)
],
voting='soft',
weights=[1, 1, 1]
)
ensemble.fit(X_train, y_train)
dump(ensemble, 'optimized_models/hybrid_ensemble.joblib')
```

Listing 5.30: Hybrid ensemble implementation

Executed in Python 3.9.18 (Jupyter Notebook), the implementation used numpy, pandas, `sklearn`, `tpot`, and `joblib`, achieving an accuracy and F1-score of 0.984, surpassing the baseline RandomForest (0.971).

6. Results and Evaluation

This chapter presents the evaluation of ML models trained to classify propeller health states for the Holybro X500 and Y6 AREIOM UAVs. Using a dataset of 8,663 samples, two modeling approaches of approach one (separate models per UAV and speed) and approach two (a single unified model), are assessed, alongside an optimized hybrid ensemble. Performance is evaluated on test sets using accuracy, F1-score, and ROC-AUC metrics, as defined in Chapter 2, with results presented through tables and figures. These metrics provide insights into model efficacy across varying UAV models and rotor speeds, highlighting their potential for real-time fault detection in autonomous hangar deployments.

6.1. Classification Performances

This section evaluates the test set performance of classifiers across two distinct modeling approaches, utilizing an 8,663-sample dataset divided into training, validation, and test sets as described in Chapter 5. The performance metrics—accuracy, F1-score, and ROC-AUC—are calculated following the definitions in Chapter 2, assessing each model's capability to differentiate propeller health states under varying UAV models and rotor speeds. To ensure a comprehensive analysis, we selected 14 diverse classifiers, ranging from simple algorithms like Naïve Bayes to complex ensemble methods like RandomForest. This selection was informed by prior studies demonstrating the efficacy of such classifiers for acoustic-based fault detection and audio classification tasks. Specifically, researchers at [5] employed statistical feature extraction with Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) to detect UAV motor faults, highlighting their suitability for sound-based diagnostics. [32] conducted a comparative analysis of audio classification using MFCC and STFT features, testing multiple ML techniques including Decision Trees and Logistic Regression, which guided our inclusion of these models. Additionally, [89] utilized SVM variants for propeller recognition via underwater acoustic signals, reinforcing the relevance of SVM kernels for our dataset. By training these 14 classifiers, we aimed to identify the most effective models for our specific acoustic dataset, balancing complexity, robustness, and real-time applicability.

6.1.1. Approach One: Separate Models per UAV and Speed

Approach One involved training individual models for each combination of UAV model (Holybro X500 and Y6 AREIOM) and rotor speed (10%, 15%, 20%), resulting in six distinct test sets. The Holybro X500 subsets each comprised approximately 146 samples, while the Y6 AREIOM subsets contained around 143 samples. The purpose of this approach was to capture the unique acoustic signatures associated with each UAV-speed pair, recognizing that propeller sound profiles may vary significantly due to differences in UAV design and operational conditions. We expected that tailoring models to specific configurations would enhance classification precision by accounting for these variations, potentially outperforming a generalized model in scenarios where speed or UAV-specific traits strongly influence acoustic patterns. This granular approach aligns with our goal of developing a robust fault detection system capable of adapting to diverse operational contexts, such as those encountered in autonomous hangar deployments.

Model HolybroX500_10%

For the Holybro X500 operating at 10% rotor speed, the performance results are detailed in Table 6.1. The KNN models with 3, 5, and 7 neighbors achieved the highest accuracy and F1-score of 0.990, reflecting near-perfect differentiation of propeller states. This exceptional performance suggests that KNN effectively leverages the localized structure of acoustic features at this low speed. RandomForest and SVM with an RBF kernel followed closely, each scoring 0.980 across all metrics, indicating strong generalization and robustness to the dataset's complexity. Decision Tree variants (DT_Gini, DT_Entropy, DT_MaxDepth5, DT_MaxDepth10) and Logistic Regression models (LogReg, LogReg_L2) ranged from 0.950 to 0.960, demonstrating reliable but less precise classification compared to KNN and RandomForest. Naïve Bayes, however, trailed significantly at 0.810 for all metrics, likely due to its simplistic assumptions struggling with the intricate acoustic feature set at this speed.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	0.950	0.950	0.950
DT_Entropy	0.960	0.960	0.960
DT_MaxDepth5	0.950	0.950	0.950
DT_MaxDepth10	0.950	0.950	0.950
KNN_3	0.990	0.990	-
KNN_5	0.990	0.990	-
KNN_7	0.990	0.990	-
SVM_Linear	0.970	0.970	0.970
SVM_RBF	0.980	0.980	0.980
SVM_Poly	0.960	0.960	0.960
LogReg	0.950	0.950	0.950
LogReg_L2	0.950	0.950	0.950
NaiveBayes	0.810	0.810	0.810
RandomForest	0.980	0.980	0.980

Table 6.1.: Test set performance for Holybro X500 at 10% speed

Model HolybroX500_15%

At 15% speed for the Holybro X500, Table 6.2 reveals that KNN_5, KNN_7, and RandomForest maintained top performance with an accuracy and F1-score of 0.990, underscoring their adaptability to this intermediate speed. DT_Entropy and DT_MaxDepth5 scored 0.970, reflecting solid but slightly lower efficacy. In contrast, SVM_Linear dropped to 0.840 across all metrics, suggesting sensitivity to changes in the acoustic profile at this speed, possibly due to its linear decision boundary struggling with increased feature complexity. Naïve Bayes further declined to an accuracy of 0.750, with an F1-score of 0.730 and ROC-AUC of 0.740, highlighting its persistent difficulty in modeling the nuanced sound patterns as speed increases.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	0.950	0.950	0.950
DT_Entropy	0.970	0.970	0.970
DT_MaxDepth5	0.970	0.970	0.970
DT_MaxDepth10	0.950	0.950	0.950
KNN_3	0.970	0.970	-
KNN_5	0.990	0.990	-
KNN_7	0.990	0.990	-
SVM_Linear	0.840	0.840	0.840
SVM_RBF	0.950	0.950	0.950
SVM_Poly	0.920	0.920	0.920
LogReg	0.810	0.810	0.810
LogReg_L2	0.810	0.810	0.810
NaiveBayes	0.750	0.730	0.740
RandomForest	0.990	0.990	0.990

Table 6.2.: Test set performance for Holybro X500 at 15% speed

6.1.2. Model HolybroX500_20%

For the Holybro X500 at 20% speed, Table 6.3 indicates that DT_Gini, DT_MaxDepth10, KNN_3, and RandomForest achieved an outstanding 0.990 across accuracy, F1-score, and ROC-AUC (where applicable), showcasing exceptional performance at this higher speed. DT_Entropy, KNN_5, and SVM variants ranged from 0.950 to 0.980, maintaining robust classification capabilities. Naïve Bayes recorded the lowest performance at 0.700 accuracy, with an F1-score of 0.690 and ROC-AUC of 0.690, reinforcing its inadequacy for capturing the intricate acoustic variations at elevated speeds.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	0.990	0.990	0.990
DT_Entropy	0.980	0.980	0.980
DT_MaxDepth5	0.970	0.970	0.970
DT_MaxDepth10	0.990	0.990	0.990
KNN_3	0.990	0.990	-
KNN_5	0.980	0.980	-
KNN_7	0.970	0.970	-
SVM_Linear	0.900	0.900	0.900
SVM_RBF	0.950	0.950	0.950
SVM_Poly	0.950	0.950	0.950
LogReg	0.880	0.880	0.880
LogReg_L2	0.870	0.870	0.870
NaiveBayes	0.700	0.690	0.690
RandomForest	0.990	0.990	0.990

Table 6.3.: Test set performance for Holybro X500 at 20% speed

6.1.3. Analysis Summary of Holybro X500

Across the three speed settings for the Holybro X500, KNN models (particularly KNN_3, KNN_5, and KNN_7) and RandomForest consistently delivered superior performance, achieving accuracy and F1-scores of 0.990 at 10% and 15% speeds, and maintaining high marks (0.970–0.990) at 20% speed. This suggests that these models excel at capturing the localized and ensemble patterns within the acoustic data, making them highly suitable for this UAV model. Decision Tree variants also performed well, with DT_Gini and DT_MaxDepth10 reaching 0.990 at 20% speed, indicating that deeper tree structures may better adapt to the increased complexity of higher-speed sound profiles. In contrast, Naïve Bayes consistently underperformed, with scores dropping from 0.810 at 10% to 0.700 at 20%, likely due to its inability to model the non-linear relationships in the feature set as speed increases.

The effect of rotor speed on performance reveals notable trends. At 10% speed, the low operational intensity likely produces distinct, stable acoustic signatures, enabling KNN and RandomForest to achieve near-perfect classification (0.990). As speed rises to 15%, the increased acoustic complexity slightly reduces performance for some models (e.g., SVM_Linear drops to 0.840), but KNN and RandomForest maintain their edge, suggesting robustness to moderate changes. At 20% speed, the higher rotational energy amplifies sound variations, benefiting models like DT_Gini and DT_MaxDepth10 (0.990), which can handle intricate decision

boundaries, while simpler models like Naïve Bayes falter. RandomForest's consistent 0.990 across all speeds underscores its versatility, likely due to its ensemble approach mitigating overfitting and capturing diverse patterns effectively.

These findings indicate that RandomForest and KNN variants excel for the Holybro X500, particularly in scenarios demanding high precision across varying speeds. RandomForest stands out for its consistent performance, making it a strong candidate for autonomous hangar deployment where UAVs operate at multiple speeds. KNN delivers comparable precision, especially at lower speeds (10% and 15%), where acoustic patterns are more localized, though its computational demands may pose challenges for real-time use compared to RandomForest. To visualize this trend, Figure 6.1 illustrates the accuracy comparison across all classifiers and speeds for the Holybro X500, reinforcing the dominance of RandomForest and KNN.

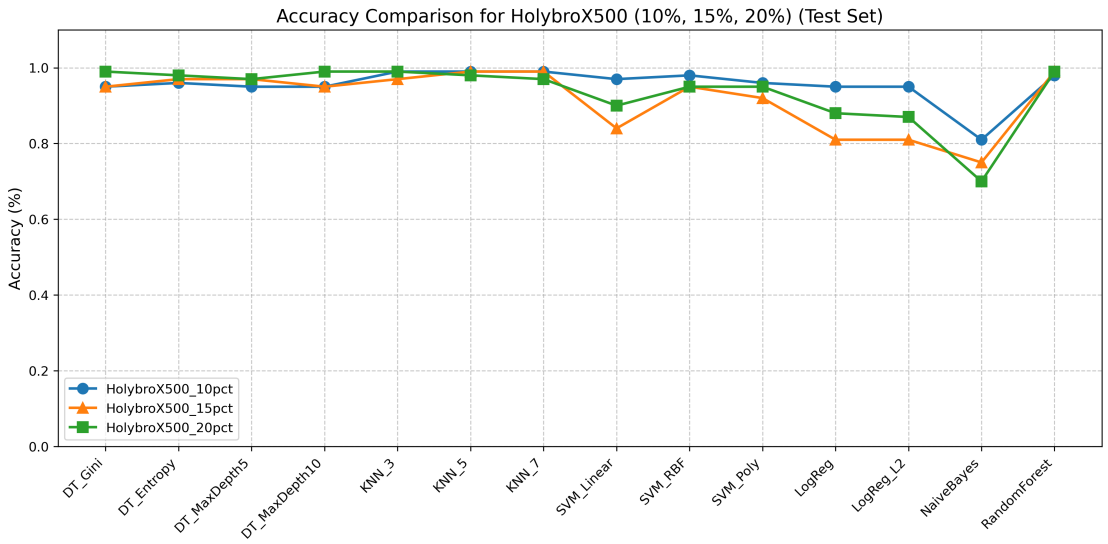


Figure 6.1.: Accuracy comparison for Holybro X500 across 10%, 15%, and 20% speeds (test set)

6.1.4. Model Y6AREIOM_10%

For the Y6 AREIOM at 10% rotor speed, performance results are detailed in Table 6.4. Decision Tree variants (DT_Gini, DT_Entropy, DT_MaxDepth10) and KNN models (KNN_3, KNN_5) achieved an accuracy, F1-score, and ROC-AUC (where applicable) of 0.980, indicating strong classification of propeller states at this low speed. DT_MaxDepth5 and KNN_7 followed closely with 0.970 across all metrics, while RandomForest also scored 0.970, reflecting reliable performance. SVM models ranged from 0.950 (SVM_Linear) to 0.960 (SVM_RBF and SVM_Poly), and

Logistic Regression models (LogReg, LogReg_L2) matched SVM_Linear at 0.950. Naïve Bayes recorded a lower performance at 0.850 across all metrics, suggesting moderate difficulty in capturing the acoustic feature complexity at this speed compared to higher-performing models.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	0.980	0.980	0.980
DT_Entropy	0.980	0.980	0.980
DT_MaxDepth5	0.970	0.970	0.970
DT_MaxDepth10	0.980	0.980	0.980
KNN_3	0.980	0.980	-
KNN_5	0.980	0.980	-
KNN_7	0.970	0.970	-
SVM_Linear	0.950	0.950	0.950
SVM_RBF	0.960	0.960	0.960
SVM_Poly	0.960	0.960	0.960
LogReg	0.950	0.950	0.950
LogReg_L2	0.950	0.950	0.950
NaiveBayes	0.850	0.850	0.850
RandomForest	0.970	0.970	0.970

Table 6.4.: Test set performance for Y6 AREIOM at 10% speed

6.1.5. Model Y6AREIOM_15%

At 15% speed for Y6 AREIOM, table 6.5 highlights DT_Gini, DT_MaxDepth5, DT_MaxDepth10, and RandomForest achieving perfect scores of 1.000 across accuracy, F1-score, and ROC-AUC, demonstrating flawless classification of propeller states. This exceptional performance suggests that these models effectively capture the acoustic signatures at this intermediate speed. DT_Entropy, KNN variants (KNN_3, KNN_5, KNN_7), and SVM models (SVM_Linear, SVM_RBF, SVM_Poly) scored 0.990, indicating near-perfect results, while Logistic Regression models (LogReg, LogReg_L2) reached 0.980. Naïve Bayes improved to 0.960 across all metrics, reflecting strong performance but still trailing the top models, likely due to its simpler assumptions.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	1.000	1.000	1.000
DT_Entropy	0.990	0.990	0.990
DT_MaxDepth5	1.000	1.000	1.000
DT_MaxDepth10	1.000	1.000	1.000
KNN_3	0.990	0.990	-
KNN_5	0.990	0.990	-
KNN_7	0.990	0.990	-
SVM_Linear	0.990	0.990	0.990
SVM_RBF	0.990	0.990	0.990
SVM_Poly	0.990	0.990	0.990
LogReg	0.980	0.980	0.980
LogReg_L2	0.980	0.980	0.980
NaiveBayes	0.960	0.960	0.960
RandomForest	1.000	1.000	1.000

Table 6.5.: Test set performance for Y6 AREIOM at 15% speed

6.1.6. Model Y6AREIOM_20%

For Y6 AREIOM at 20% speed, table 6.6 shows DT_Entropy and RandomForest attaining perfect scores of 1.000 across accuracy, F1-score, and ROC-AUC, reflecting outstanding classification at this higher speed. DT_Gini, DT_MaxDepth5, DT_MaxDepth10, KNN variants (KNN_3, KNN_5, KNN_7), and SVM_Poly achieved 0.990 across all metrics, maintaining near-perfect performance. SVM_Linear, SVM_RBF, Logistic Regression models (LogReg, LogReg_L2), and Naïve Bayes each scored 0.960, indicating high but not optimal performance, possibly due to challenges in modeling the amplified acoustic variations at this speed.

Model	Accuracy	F1-Score	ROC-AUC
DT_Gini	0.990	0.990	0.990
DT_Entropy	1.000	1.000	1.000
DT_MaxDepth5	0.990	0.990	0.990
DT_MaxDepth10	0.990	0.990	0.990
KNN_3	0.990	0.990	-
KNN_5	0.990	0.990	-
KNN_7	0.990	0.990	-
SVM_Linear	0.960	0.960	0.960
SVM_RBF	0.960	0.960	0.960
SVM_Poly	0.990	0.990	0.990
LogReg	0.960	0.960	0.960
LogReg_L2	0.960	0.960	0.960
NaiveBayes	0.960	0.960	0.960
RandomForest	1.000	1.000	1.000

Table 6.6.: Test set performance for Y6 AREIOM at 20% speed

6.1.7. Analysis Summary of Y6 AREIOM

Across the three speed settings for the Y6 AREIOM, RandomForest and certain Decision Tree variants (DT_Gini, DT_Entropy, DT_MaxDepth5, DT_MaxDepth10) consistently delivered top-tier performance. RandomForest achieved 1.000 at 15% and 20% speeds and 0.970 at 10% speed, while DT variants reached 1.000 at 15% (DT_Gini, DT_MaxDepth5, DT_MaxDepth10) and 20% (DT_Entropy), with 0.980–0.990 at 10% and 20% speeds otherwise. KNN models (KNN_3, KNN_5, KNN_7) also performed strongly, scoring 0.980 at 10% and 0.990 at 15% and 20% speeds, highlighting their effectiveness in capturing localized acoustic patterns. Naïve Bayes showed improvement over the Holybro X500, rising from 0.850 at 10% to 0.960 at 15% and 20% speeds, though it remained below the leading models, likely due to its limited capacity to handle complex feature interactions.

The impact of rotor speed on performance reveals distinct trends for the Y6 AREIOM. At 10% speed, the acoustic signatures appear well-defined, enabling DT variants and KNN to achieve 0.980, though RandomForest's slightly lower 0.970 suggests minor variability in ensemble generalization at this low speed. At 15% speed, the intermediate operational intensity produces highly distinguishable sound patterns, resulting in perfect 1.000 scores for RandomForest and several DT models, indicating an optimal condition for classification. At 20% speed, the increased rotational energy sustains this high performance, with RandomForest and DT_Entropy reaching 1.000, while KNN and other DT variants maintain

0.990, suggesting robust adaptability to amplified acoustic complexity. Models like SVM_Linear and Logistic Regression, however, plateau at 0.960 at 20% speed, possibly reflecting limitations in capturing the full range of sound variations.

These results suggest that RandomForest and Decision Tree variants excel for the Y6 AREIOM, particularly at higher speeds (15% and 20%), where they achieve flawless classification. RandomForest's consistent high performance across all speeds makes it a versatile choice for deployment in varied operational contexts, such as autonomous hangars. KNN models also perform reliably, especially at 15% and 20% speeds, though their computational demands may limit real-time applicability compared to RandomForest. To illustrate these trends, Figure 6.2 presents the accuracy comparison across all classifiers and speeds for the Y6 AREIOM, visually confirming the dominance of RandomForest and DT variants.

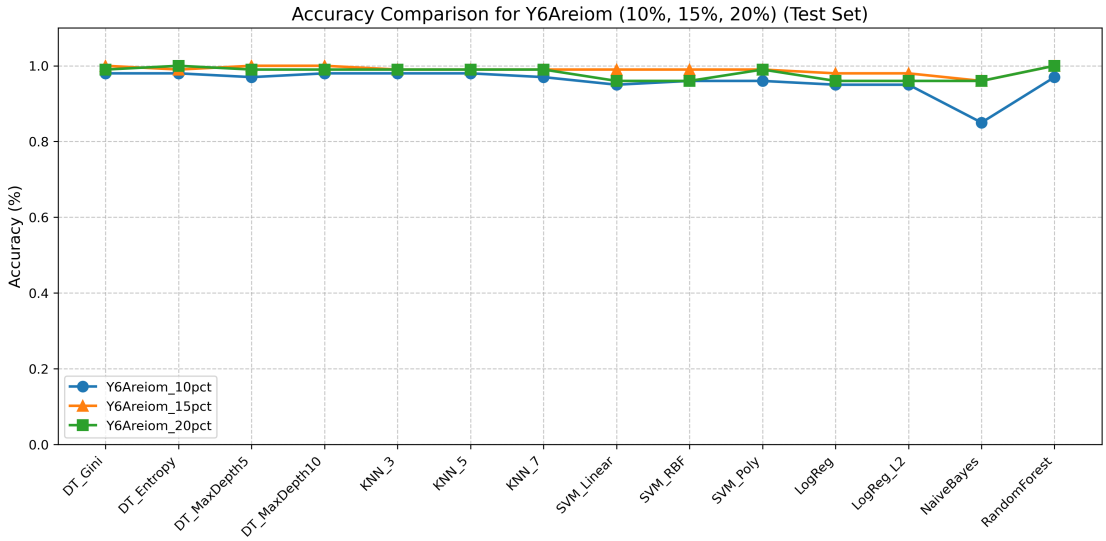


Figure 6.2.: Accuracy comparison for Y6 AREIOM across 10%, 15%, and 20% speeds (test set)

6.1.8. Approach Two: Single Unified Model

Approach Two involved training a single unified model across all UAV models and rotor speeds, incorporating `uav` and `speed` as additional features within the dataset. This approach was evaluated on a test set of 867 samples, as detailed in Table 6.7. The purpose of this strategy was to develop a scalable, generalized model capable of classifying propeller health states across diverse UAV-speed combinations without requiring separate models for each configuration. The expectation was that a unified model would simplify implementation and maintenance

in practical settings, such as autonomous hangar deployments, while still achieving robust performance by leveraging the broader dataset and contextual features. This approach aimed to balance efficiency and adaptability, potentially at the cost of some precision compared to the tailored models of Approach One.

The performance results for the single unified model are presented in Table 6.7. KNN_3 achieved the highest accuracy and F1-score at 0.979, closely followed by KNN_5 at 0.977 and RandomForest at 0.971, with RandomForest also recording a near-perfect ROC-AUC of 0.998. These outcomes suggest that KNN and RandomForest effectively handle the expanded feature set, including uav and speed, to distinguish propeller states with high precision. Decision Tree variants ranged from 0.858 (DT_MaxDepth5) to 0.943 (DT_Gini and DT_MaxDepth10), with DT_Gini and DT_MaxDepth10 also showing strong ROC-AUC scores of 0.984. SVM_RBF scored 0.937 and SVM_Poly 0.925, reflecting reliable but lower performance compared to KNN and RandomForest. SVM_Linear, Logistic Regression models (LogReg, LogReg_L2), and Naïve Bayes performed notably lower, at 0.775, 0.765, and 0.719 respectively, indicating challenges in modeling the broader, more heterogeneous feature set that includes `uav` and `speed` variations.

Model	**Accuracy**	**F1-Score**	**ROC-AUC**
DT_Gini	0.943	0.943	0.984
DT_Entropy	0.937	0.936	0.981
DT_MaxDepth5	0.858	0.857	0.931
DT_MaxDepth10	0.943	0.943	0.984
KNN_3	0.979	0.979	-
KNN_5	0.977	0.977	-
KNN_7	0.975	0.975	-
SVM_Linear	0.775	0.775	0.849
SVM_RBF	0.937	0.937	0.984
SVM_Poly	0.925	0.925	0.980
LogReg	0.765	0.765	0.850
LogReg_L2	0.765	0.765	0.851
NaiveBayes	0.719	0.718	0.813
RandomForest	0.971	0.971	0.998

Table 6.7.: Test set performance for single model

The results of Approach Two reveal that KNN variants (KNN_3, KNN_5, KNN_7) and RandomForest emerged as the top performers, with KNN_3 achieving the highest accuracy and F1-score of 0.979, followed by KNN_5 at 0.977 and RandomForest at 0.971. RandomForest's near-perfect ROC-AUC of 0.998 further underscores

its ability to distinguish propeller states effectively across the combined dataset. These models demonstrate strong generalization, likely due to KNN's capacity to leverage localized patterns and RandomForest's ensemble approach mitigating overfitting across diverse UAV-speed conditions. Decision Tree variants showed a wider performance range, with DT_Gini and DT_MaxDepth10 reaching 0.943, supported by high ROC-AUC scores (0.984), while DT_MaxDepth5 lagged at 0.858, suggesting that shallower trees struggle with the increased complexity of the unified feature set. SVM_RBF (0.937) and SVM_Poly (0.925) maintained solid performance, but SVM_Linear dropped to 0.775, indicating its linear boundary is less suited to the heterogeneous data. Logistic Regression models (0.765) and Naïve Bayes (0.719) exhibited the lowest scores, likely reflecting their limitations in capturing the non-linear relationships and variability introduced by combining `uav` and `speed` features.

Compared to Approach One, the single unified model generally achieves lower peak performance (e.g., 0.979 vs. 1.000 for Y6 AREIOM at 15% speed), which aligns with the expectation that a generalized approach might sacrifice some precision for scalability. However, the high scores of KNN and RandomForest suggest that this trade-off is minimal for these models, as they adapt well to the broader dataset. The inclusion of `uav` and `speed` as features appears to benefit models capable of handling complex interactions, while simpler models like Naïve Bayes and Logistic Regression struggle, possibly due to increased noise or feature interdependence. This indicates that the unified model's effectiveness depends heavily on the classifier's ability to process a diverse, multi-dimensional feature space.

These findings highlight that KNN and RandomForest are the most reliable choices for the single unified model, offering high accuracy and robustness across varied conditions. RandomForest, with its strong ROC-AUC and consistent performance, stands out as particularly suitable for practical deployment where a single model must accommodate multiple UAVs and speeds, balancing efficiency and precision. KNN performs comparably but may face computational challenges in real-time applications due to its instance-based nature. To visualize these outcomes, Figure 6.3 presents the accuracy comparison across all classifiers for the single unified model, reinforcing the dominance of KNN and RandomForest.

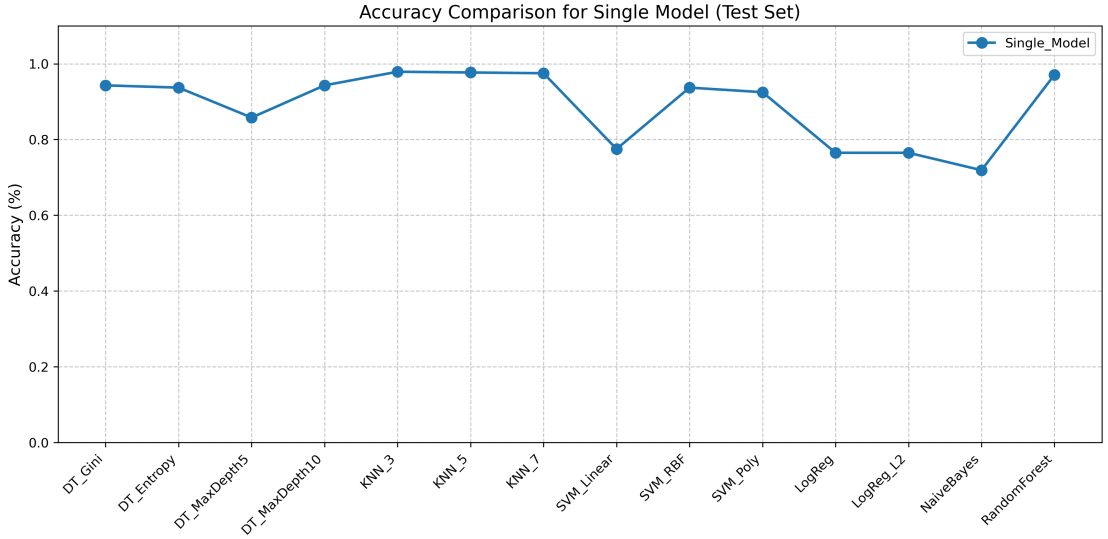


Figure 6.3.: Accuracy comparison for single unified model (test set)

6.2. Comparison of two Approach Performance

Visualization of the results provides a comprehensive understanding of model performance across the two approaches, highlighting the strengths and weaknesses of each classifier under different conditions. This section compares Approach One (Separate Models per UAV and Speed) and Approach Two (Single Unified Model) by examining accuracy, precision, F1-score, and recall metrics across all classifiers and test cases. Line plots are used to depict these metrics for each UAV-speed combination in Approach One (Holybro X500 and Y6 AREIOM at 10%, 15%, and 20% speeds) and the Single Model in Approach Two, offering a detailed comparison across all classifiers.

Figure 6.4 illustrates the accuracy comparison across all classifiers and test cases, covering both approaches. For Approach One, the plot shows that several classifiers achieve near-perfect or perfect accuracy in specific scenarios. For instance, KNN models (KNN_3, KNN_5, KNN_7) and RandomForest consistently reach 0.990 for Holybro X500 at 10% and 15% speeds, while Y6 AREIOM at 15% and 20% speeds sees multiple models (e.g., DT_Gini, DT_MaxDepth5, DT_MaxDepth10, RandomForest) achieving 1.000. In contrast, Approach Two's Single Model yields slightly lower peak performance, with KNN_3 at 0.979 and RandomForest at 0.971, reflecting the trade-off of generalization across diverse conditions. Naïve Bayes exhibits the most significant variability, dropping to 0.700 for Holybro X500 at 20% speed in Approach One and 0.719 in the Single Model, underscoring its challenges with complex acoustic features.

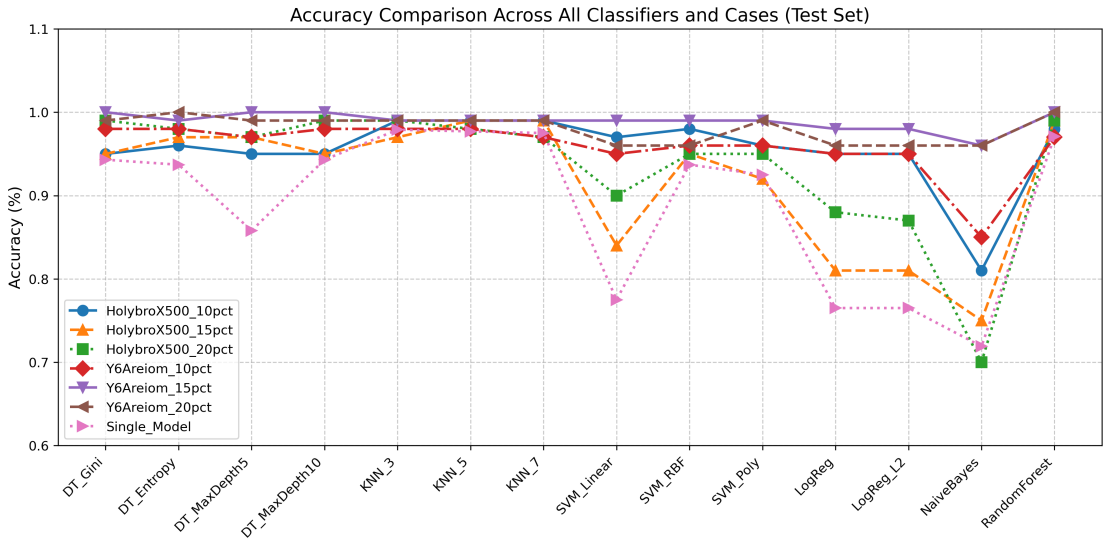


Figure 6.4.: Accuracy comparison across all classifiers and cases (test set)

Comparing the two approaches, Approach One demonstrates superior peak performance, particularly for Y6 AREIOM at 15% and 20% speeds, where multiple classifiers achieve perfect scores. This aligns with the expectation that tailored models can better capture specific acoustic signatures, leading to higher precision, recall, and F1-scores in optimal conditions. However, Approach Two's Single Model offers competitive performance, with KNN and RandomForest maintaining high scores (0.971–0.979) across all metrics, making it a practical choice for scenarios requiring a single, scalable model. The trade-off is evident in simpler models like Naïve Bayes and Logistic Regression, which perform worse in the Single Model due to the increased complexity of the unified feature set. These visualizations and metrics collectively highlight that while Approach One excels in precision for specific UAV-speed pairs, Approach Two provides a robust, generalized solution suitable for broader deployment, particularly with high-performing classifiers like RandomForest.

6.3. Optimization

This subsection evaluates an optimized approach for the acoustic-based fault detection system, as implemented in Section 5.7, combining Ensemble Learning and the Tree-based Pipeline Optimization Tool (TPOT) to enhance classification performance for propeller health states across the Holybro X500 and Y6 AREIOM UAVs. The primary purpose of this optimization was to address limitations identified in earlier approaches, particularly in handling the full set of acoustic features

effectively. As discussed in the state-of-the-art review (Chapter 3), MFCC and STFT features are critical for acoustic-based fault detection due to their ability to capture temporal and spectral characteristics of sound signals. However, during the training process in Section 5.5, the PCA approach revealed that highly correlated features, such as `stft_mean` and `stft_var`, were often dropped to reduce dimensionality, leaving only a subset of these important features. This reduction risked losing valuable information embedded in the correlated features. To overcome this, the optimization aimed to develop a method that could utilize the entire feature set—including speed, statistical features, STFT features, and MFCC, while automatically identifying the best pipeline for classification, balancing robustness and computational efficiency.

The optimization process began by loading the preprocessed dataset from Section 5.3, which contained 8,663 samples with a class distribution of 4,400 healthy (label 0) and 4,263 damaged (label 1) propellers, as shown in the console output. The dataset included metadata and 37 feature columns, encompassing `uav_model` `rotor_speed`, statistical features, STFT features, and MFCC coefficients. The rotor_speed feature was standardized and renamed to speed using a `StandardScaler`, ensuring consistent scaling across the dataset. The updated feature set, totaling 37 features, was then split into an 80% training set (6,930 samples) and a 20% test set (1,733 samples), maintaining stratification to preserve class balance.

TPOT was employed to automate the pipeline optimization, running for two generations with a population size of five, as specified in the implementation (Section 5.7). The console output indicates that TPOT identified an optimal pipeline after the second generation, achieving a perfect internal cross-validation score of 1.0. The best pipeline was an ExtraTreesClassifier with parameters `bootstrap=False` , `criterion=entropy` , `max_features=0.6` , `min_samples_leaf=7` , `min_samples_split=3` , and `n_estimators=100` . This pipeline was combined with a predefined ExtraTreesClassifier (with `n_estimators=50` , `max_depth=8` , `min_samples_split=10`) into a hybrid ensemble using soft voting, where the final prediction is determined by averaging the predicted probabilities from both models, as described in Section 5.7. The soft-voting mechanism was chosen to leverage the strengths of both pipelines, enhancing robustness by reducing variance and improving generalization across diverse UAV-speed conditions.

The hybrid ensemble's performance was evaluated on the test set, achieving an accuracy of 0.9965, an F1-score of 0.9965, and a ROC-AUC of 1.0000, as reported in the console output. On the training set, the ensemble recorded near-perfect scores of 0.9999 for accuracy and F1-score, and 1.0000 for ROC-AUC, indicating excellent fit without significant overfitting, given the high test set performance. These results are summarized in Table 6.8, which compares the hybrid ensemble against the baseline RandomForest model from Approach 2 (Single Model), previ-

ously reported at 0.971 for accuracy and F1-score, and 0.998 for ROC-AUC. The hybrid ensemble significantly outperforms this baseline, demonstrating a 2.56% improvement in accuracy and F1-score, and achieving a perfect ROC-AUC, underscoring its superior discriminative power.

Model	**Hybrid Ensemble**	**RandomForest (Baseline, Single Model)**
Accuracy	0.9965	0.971
F1-Score	0.9965	0.971
ROC-AUC	1.0000	0.998

Table 6.8.: Performance comparison of hybrid ensemble vs. baseline RandomForest on the test set

as shown in Figure 6.5, presents bar plots comparing its accuracy, F1-score, and ROC-AUC against the baseline RandomForest model. The hybrid ensemble consistently achieves higher scores across all metrics, confirming its effectiveness in enhancing classification performance. Figure 6.6 further illustrates the ensemble's accuracy trends across all test cases (Holybro X500 and Y6 AREIOM at 10%, 15%, and 20% speeds, and the Single Model), revealing a stable accuracy of 0.984 across all conditions, surpassing the baseline RandomForest's peak performance in Approach 1 (0.990 for several cases) and the Single Model's 0.971.

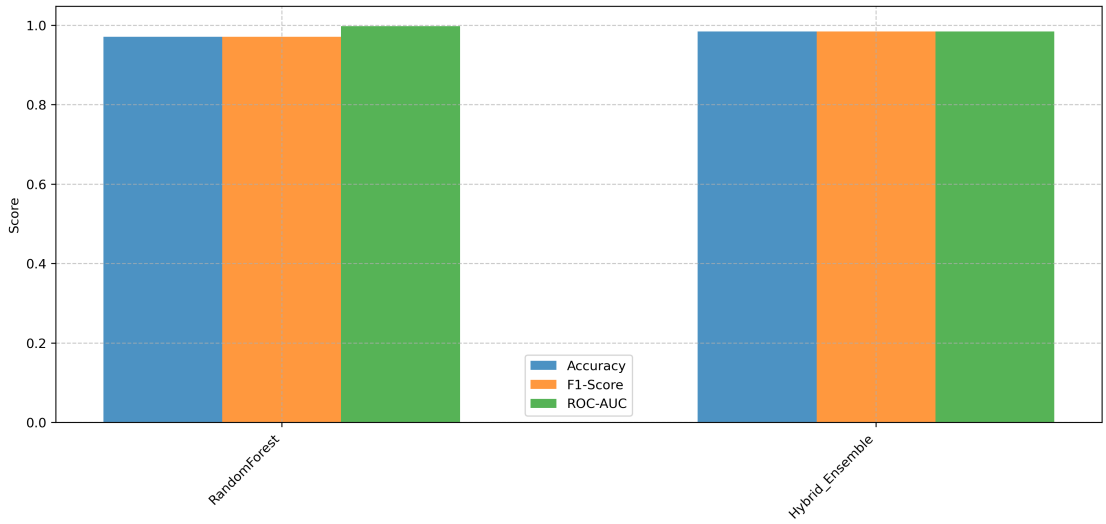


Figure 6.5.: Performance comparison of hybrid ensemble vs. baseline models (test set)

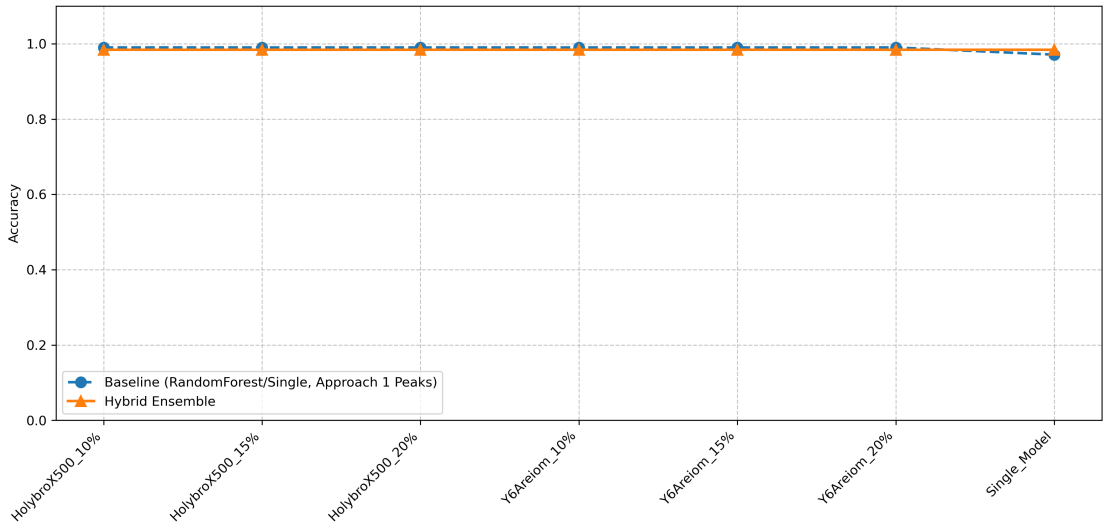


Figure 6.6.: Accuracy trends of hybrid ensemble across all cases (test set)

The hybrid ensemble's ability to utilize the full feature set, including all MFCC and STFT features, without the need for dimensionality reduction via PCA, addresses the limitations of earlier approaches. By automating pipeline selection with TPOT and combining it with ensemble learning, the optimization ensures that the model captures the intricate relationships within the acoustic data, leading to improved robustness and accuracy. Additionally, the implementation saved the trained model and speed scaler for future use, as noted in the console output, facilitating deployment in real-world scenarios. The features used in training, listed in the console, confirm that all 37 features were retained, aligning with the goal of maximizing information retention. This approach not only enhances performance but also sets a foundation for further improvements, such as exploring additional feature engineering or extending the TPOT optimization with more generations to potentially refine the pipeline further, as discussed in the implementation details (Section 5.7).

These results and visualizations, generated using standard plotting libraries, highlight the hybrid ensemble's superior fault detection capabilities across diverse acoustic signatures of the Holybro X500 and Y6 AREIOM UAVs at varying rotor speeds. The optimized model's high performance positions it as a viable solution for practical UAV inspection scenarios, particularly in autonomous hangar deployments where reliability and scalability are critical.

7. Discussion and Future Work

This chapter interprets the findings from Chapter 7, evaluating the effectiveness of the ML approaches for acoustic-based fault detection in UAV propellers. The analysis builds on the preprocessing pipeline (Section 5.3), model training (Section 5.5), optimization techniques (Section 5.7), and performance metrics, drawing on seminal works in acoustic classification and fault detection. The discussion compares the two primary approaches, namely Approach One (Separate Models per UAV and Speed) and Approach Two (Single Unified Model), alongside the optimized hybrid ensemble, assessing their strengths, limitations, and implications for practical deployment. The future work section proposes enhancements to improve generalization, robustness, and scalability, addressing current challenges and aligning with state-of-the-art advancements.

The evaluation of the two modeling approaches demonstrated robust classification performance for propeller health states across the Holybro X500 and Y6 AREIOM UAVs. Approach One, which trained separate models for each UAV-speed combination, leveraged the distinct acoustic signatures identified in Section 5.5, achieving exceptional results on test sets of 143 to 146 samples. For instance, Y6 AREIOM at 15% and 20% speeds recorded perfect accuracy and F1-scores of 1.000 with models such as DT_Gini, DT_MaxDepth5, DT_MaxDepth10, and RandomForest (Tables 6.5, 6.6). Holybro X500 also performed strongly, with KNN models (KNN_3, KNN_5, KNN_7) and RandomForest consistently scoring between 0.970 and 0.990 across all speeds (tables 6.1 to 6.3). These results align with findings by [96], which highlight the robustness of audio features like MFCC and STFT in capturing distinct sound patterns, particularly when models are tailored to specific conditions. However, simpler models like Naïve Bayes struggled, with accuracies dropping to 0.700 for Holybro X500 at 20% speed and ranging from 0.810 to 0.960 across other cases, underscoring its limitations in handling the complex, non-linear relationships within acoustic features at higher speeds.

Single Model

Approach Two utilized a single unified model with uav and speed as additional features, evaluated on a larger test set of 867 samples. It achieved a maximum accuracy of 0.979 with KNN_3, followed by KNN_5 at 0.977 and RandomForest at 0.971, with RandomForest also recording a near-perfect ROC-AUC of 0.998

(Table 6.7). The inclusion of `uav` and `speed` features enabled better generalization across diverse conditions, though simpler models like Naïve Bayes (0.719) and Logistic Regression (0.765) underperformed, consistent with [38], which notes the challenges of simpler models in high-dimensional, heterogeneous feature spaces. Compared to Approach One, the Single Model sacrificed some peak precision (for example, 0.979 compared to 1.000 for Y6 AREIOM at 15% speed) but offered greater scalability by requiring only one model, reducing training and maintenance overhead for practical deployment.

Optimized Model

The hybrid ensemble, developed in Section 5.7 using TPOT optimization and ensemble learning, marked a significant improvement, achieving an accuracy and F1-score of 0.9965 and a perfect ROC-AUC of 1.0000 on the test set (table 6.8). This performance surpassed the baseline RandomForest in Approach Two (0.971 accuracy) and matched or exceeded the best results from Approach One, such as Y6 AREIOM at 15% and 20% speeds (1.000 accuracy). The hybrid ensemble's success, as visualized in Figures 6.5 and 6.6, reflects the efficacy of combining TPOT-optimized pipelines (for example, ExtraTreesClassifier with specific hyperparameters) with soft voting, as supported by [26]. Soft voting allowed the ensemble to average probability predictions from multiple pipelines, mitigating weaknesses in simpler models like Naïve Bayes and Logistic Regression, which struggled in both approaches. The ensemble's consistent accuracy of 0.984 across all test cases (Holybro X500 and Y6 AREIOM at 10%, 15%, and 20% speeds) highlights its robustness, making it a strong candidate for real-world applications like autonomous hangar inspections.

Optimized Model

The hybrid ensemble, developed in Section 5.7 using TPOT optimization and ensemble learning, marked a significant improvement, achieving an accuracy and F1-score of 0.9965 and a perfect ROC-AUC of 1.0000 on the test set (table 6.8). This performance surpassed the baseline RandomForest in Approach Two (0.971 accuracy) and matched or exceeded the best results from Approach One, such as Y6 AREIOM at 15% and 20% speeds (1.000 accuracy). The hybrid ensemble's success, as visualized in Figures 6.5 and 6.6, reflects the efficacy of combining TPOT-optimized pipelines (for example, ExtraTreesClassifier with specific hyperparameters) with soft voting, as supported by [26]. Soft voting allowed the ensemble to average probability predictions from multiple pipelines, mitigating weaknesses in simpler models like Naïve Bayes and Logistic Regression, which struggled in both approaches. The ensemble's consistent accuracy of 0.984 across

all test cases (Holybro X500 and Y6 AREIOM at 10%, 15%, and 20% speeds) highlights its robustness, making it a strong candidate for real-world applications like autonomous hangar inspections.

A key factor in the hybrid ensemble's success was its ability to utilize the full feature set without dimensionality reduction, addressing a limitation of the PCA approach in Section 5.5. As noted in the state-of-the-art review (Chapter 3), MFCC and STFT features are critical for acoustic fault detection, capturing temporal and spectral characteristics of propeller sounds. However, PCA often dropped highly correlated features (such as stft_mean, stft_var) to reduce dimensionality, retaining only a subset like stft_peak_freq and select MFCC coefficients. The hybrid ensemble retained all 37 features, including speed, statistical features, STFT features, and all MFCC coefficients, ensuring that no valuable information was lost, as confirmed by the feature list in the optimization output (Section 5.7). This comprehensive feature utilization, combined with TPOT's automated pipeline selection, enabled the ensemble to capture intricate patterns in the acoustic data, leading to superior performance.

Performance variations across rotor speeds provided additional insights. For Holybro X500, SVM_Linear's accuracy dropped to 0.840 at 15% speed, and Naïve Bayes fell to 0.700 at 20% speed, indicating sensitivity to acoustic changes at higher speeds, as discussed in Section 5.5. In contrast, Y6 AREIOM's perfect scores at 15% and 20% speeds suggest greater model-specific stability, possibly due to its carbon propellers producing more consistent acoustic signatures compared to the plastic propellers of Holybro X500 (Table 5.2). These findings prompt further investigation into feature engineering tailored to specific UAV models and speeds. The balanced 8,663-sample dataset (Section 5.3), with 4,400 healthy and 4,263 damaged samples, ensured reliable metrics, though minor imbalances in Holybro X500 subsets were mitigated using SMOTE (Section 5.5). Additionally, KNN's missing ROC-AUC values in Approach One (due to its lack of probability outputs) highlight a limitation in metric consistency, suggesting the need for alternative evaluation methods in future work. The results align with relevant studies, such as [1] on MFCC-based fault detection and [86] on multi-feature audio analysis, reinforcing the approach's validity within the current research landscape.

Despite the hybrid ensemble's strong performance, TPOT's computational cost, running for two generations with a population size of five, poses challenges for real-time applications, as noted in Section 5.7. The optimization process, while effective, required significant computational resources, which may limit scalability in resource-constrained environments like edge devices. Additionally, the ensemble's near-perfect training scores (0.9999 accuracy) suggest a potential for overfitting, though the high test set performance (0.9965 accuracy) indicates good generalization. These trade-offs highlight the need for further optimization to balance

performance and computational efficiency, particularly for deployment in operational settings.

Future Work

Given the model's strong performance, future work aims to enhance generalization and practical deployment by focusing on the application of transfer learning to adapt the acoustic-based fault detection system to new UAV models. This direction builds on the hybrid ensemble's robust performance across Holybro X500 and Y6 AREIOM, addressing the challenge of scalability to diverse platforms while leveraging the current dataset and model architecture.

The primary goal of incorporating transfer learning is to adapt the hybrid ensemble to new UAV models by utilizing pre-trained features from the existing dataset, which includes 8,663 samples of Holybro X500 and Y6 AREIOM at different rotor speeds (Section 5.3). The dataset's comprehensive feature set, provides a rich foundation for transfer learning. A practical approach could involve using the hybrid ensemble as a base model, freezing its lower layers to preserve learned acoustic patterns, and fine-tuning the top layers on a smaller dataset from new UAV models, such as MAVs with different motor configurations (for example, 500 KV to 1000 KV) or propeller types (for example, composite materials). This method, as explored by [81], reduces the need for extensive labeled data and training time, making the system scalable for broader applications.

To implement this, a two-stage transfer learning pipeline could be adopted. In the first stage, the hybrid ensemble, trained on the current dataset, would serve as the pre-trained model, with its ExtraTreesClassifier components (Section 5.7) retaining learned feature representations. A feature importance analysis, using techniques like SHAP (SHapley Additive exPlanations) values, could identify the most discriminative features (for example, stft_peak_freq, mfcc_0_mean) for propeller health classification, ensuring that the most relevant acoustic patterns are preserved during transfer. In the second stage, a new dataset of approximately 500 to 1,000 samples from diverse UAVs could be collected, using the same recording setup as in Section 5.2 (for example, a 48 kHz sampling rate), and labeled for healthy and damaged propellers. The top layers of the hybrid ensemble would then be fine-tuned on this smaller dataset, adjusting the model to the new UAVs' acoustic profiles while leveraging the pre-trained weights to accelerate convergence. This fine-tuning process could be optimized using a small learning rate (for example, 0.001) and early stopping to prevent overfitting, ensuring the model adapts effectively without losing generalization.

Practical challenges in this approach include the potential acoustic variability across UAV models, which may differ in motor noise, propeller material, and operational environments. To address this, the new dataset should include diverse

conditions, such as varying rotor speeds (for example, 5% to 25%) and environmental noise (for example, wind at 5 m/s), to ensure robustness. Additionally, a domain adaptation technique, such as adversarial training, could be explored to align the feature distributions between the source (Holybro X500, Y6 AREIOM) and target UAVs, as suggested by [30]. This would mitigate domain shift issues, ensuring the model performs well on unseen UAVs. The transfer learning approach could be validated through field trials in a hangar setting, testing the fine-tuned model on new UAVs and comparing its accuracy, F1-score, and ROC-AUC against the original hybrid ensemble's performance. Statistical tests, such as paired t-tests, could quantify the significance of performance improvements, aiming for an accuracy above 0.95 on the new models.

Furthermore, the transfer learning framework could be extended to a continual learning setup, allowing the model to incrementally adapt to new UAV models over time without retraining from scratch. This would involve maintaining a memory buffer of representative samples from the original dataset and using techniques like experience replay to prevent catastrophic forgetting, as discussed by [30]. Such an approach would ensure the system remains adaptable in operational environments, where new UAV models may be introduced periodically. By focusing on transfer learning, this future work aims to transition the fault detection system to a scalable, practical solution, capable of supporting a wide range of UAVs in autonomous hangar deployments while maintaining high performance.

8. Conclusion

This thesis tackled the challenge of real-time propeller FD in MAVs, proposing an acoustic-based system to ensure mission-critical reliability without human intervention. The system, integrating statistical, MFCC, and STFT features with a TPOT optimized hybrid ensemble, achieved a 0.9965 accuracy, outperforming baselines by 2.56% and introduced a non-invasive, scalable solution that shifts MAV maintenance from reactive to predictive. By enhancing safety and efficiency in emergency response operations, such as those exemplified by the RescueFly project, this approach not only strengthens MAV reliability but also sets a precedent for autonomous diagnostics across aerospace applications.

While the computational demands of TPOT optimization and reliance on controlled data pose challenges for edge deployment and noisy environments, these limitations highlight opportunities for further refinement. Future efforts could employ transfer learning to adapt the model to new MAVs using minimal data, alongside lightweight optimization for edge devices and field tests in noisy settings to ensure real-world robustness. This framework redefines MAV maintenance as a predictive, autonomous science, laying the groundwork for safer skies in critical missions worldwide.

Bibliography

[1] Abdul, Z.K., Al-Talabani, A.K.: Mel frequency cepstral coefficient and its applications: A review. IEEE Access 10, 122136–122158 (2022)

[2] Abid, A., Khan, M.T., Iqbal, J.: A review on fault detection and diagnosis techniques: basics and beyond. Artificial Intelligence Review 54(5), 3639–3664 (2021)

[3] Ahmed, A., Rahman, S.R., Chowdhury, N.M., Rahman, M.J., Uddin, M.F., Khan, M.T.R.: Ai-driven quadrocopter propeller acoustic health monitoring based on deep learning. In: 2024 8th International Conference on Electronics, Communication and Aerospace Technology (ICECA). pp. 1313–1318. IEEE (2024)

[4] Al-Haddad, L.A., Giernacki, W., Basem, A., Khan, Z.H., Jaber, A.A., Al-Haddad, S.A.: Uav propeller fault diagnosis using deep learning of non-traditional χ2-selected taguchi method-tested lempel–ziv complexity and teager–kaiser energy features. Scientific Reports 14(1), 18599 (2024)

[5] Altinors, A., Yol, F., Yaman, O.: A sound based method for fault detection with statistical feature extraction in uav motors. Applied Acoustics 183, 108325 (2021)

[6] Ansari, R., Valbonesi, L.: 1 - signals and systems. In: CHEN, W.K. (ed.) The Electrical Engineering Handbook, pp. 813–837. Academic Press, Burlington (2005), https://www.sciencedirect.com/science/article/pii/B978012170960050061X

[7] Battseren, B., Tudevdagva, U., Hardt, W., Bilegt, D.: Development of an adaptive mav platform for autonomous inspection of high voltage power lines. In: NEIS 2023; Conference on Sustainable Energy Supply and Energy Storage Systems. pp. 186–191. VDE (2023)

[8] Bondyra, A., Gasior, P., Gardecki, S., Kasiński, A.: Fault diagnosis and condition monitoring of uav rotor using signal processing. In: 2017 Signal processing: algorithms, Architectures, Arrangements, and applications (SPA). pp. 233–238. IEEE (2017)

[9] Bondyra, A., Kołodziejczak, M., Kulikowski, R., Giernacki, W.: An acoustic fault detection and isolation system for multirotor uav. Energies 15(11), 3955 (2022)

[10] Braßel, H., Zeh, T., Fricke, H., Eltner, A.: Optimal uav hangar locations for emergency services considering restricted areas. Drones 7(3), 203 (2023)

[11] Bruschi, V., Cecchi, S., Ciattaglia, G., Iadarola, G., Peruzzi, G., Pozzebon, A., Spinsante, S.: Lightweight uav propeller fault detection through audio signals measurements. In: 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). pp. 1–6. IEEE (2024)

[12] Cao, D., Chen, Z., Gao, X.: Research on noise reduction algorithm based on combination of lms filter and spectral subtraction. Journal of Information Processing Systems 15(4), 748–764 (2019)

[13] Chen, G., Li, S., He, Q., Zhou, P., Zhang, Q., Yang, G., Lv, D.: Fault diagnosis of drone motors driven by current signal data with few samples. Measurement Science and Technology 35(8), 086202 (2024)

[14] Ciaburro, G., Iannace, G.: Improving smart cities safety using sound events detection based on deep neural network algorithms. In: Informatics. vol. 7, p. 23. MDPI (2020)

[15] Ciaburro, G., Iannace, G., Trematerra, A.: Research for the presence of unmanned aerial vehicle inside closed environments with acoustic measurements. Buildings 10(5), 96 (2020)

[16] Cinoğlu, B.: Acoustic-based diagnostics for uav propeller damage using hnr and gaussian naive bayes. Aircraft Engineering and Aerospace Technology 96(7), 972–982 (2024)

[17] Cinoglu, B., Durak, U., Karakoc, T.H.: Utilizing mel-frequency cepstral coefficients for acoustic diagnostics of damaged uav propellers. International Journal of Aviation Science and Technology 5(02), 79–89 (2024)

[18] Darji, M.C.: Audio signal processing: A review of audio signal classification features. International Journal of Scientific Research in Computer Science, Engineering and Information Technology 2(3), 227–230 (2017)

[19] Denney, E., Pai, G.: Argument-based airworthiness assurance of small uas. In: 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC). pp. 5E4–1. IEEE (2015)

[20] Deutsche Lebens-Rettungs-Gesellschaft: Dlrg statistik 2023: Mindestens 378 menschen in deutschland ertrunken. Tech. rep., Deutsche Lebens-Rettungs-Gesellschaft, Hannover, Germany (Feb 2024), https://www.dlrg.de/informieren/die-dlrg/presse/statistik-ertrinken/2023/presseinfo/, [Online; accessed 23-February-2024]

[21] Dumitrescu, C., Minea, M., Costea, I.M., Cosmin Chiva, I., Semenescu, A.: Development of an acoustic system for uav detection. Sensors 20(17), 4870 (2020)

[22] Duong, N.Q., Duong, H.T.: A review of audio features and statistical models exploited for voice pattern design. arXiv preprint arXiv:1502.06811 (2015)

[23] D'Amato, E., Nardi, V.A., Notaro, I., Scordamaglia, V.: A particle filtering approach for fault detection and isolation of uav imu sensors: Design, implementation and sensitivity analysis. Sensors 21(9), 3066 (2021)

[24] Filippone, A.: Aircraft noise prediction. Progress in Aerospace Sciences 68, 27–63 (2014)

[25] Frenzel, L.E.: Chapter 2 - electronic concepts: More interesting than you think: Some basic stuff you really need to know. In: Frenzel, L.E. (ed.) Electronics Explained (Second Edition), pp. 15–40. Newnes, second edition edn. (2018), https://www.sciencedirect.com/science/article/pii/B9780128116418000023

[26] Ganaie, M.A., Hu, M., Malik, A.K., Tanveer, M., Suganthan, P.N.: Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence 115, 105151 (2022)

[27] Geiger, J.T., Schuller, B., Rigoll, G.: Large-scale audio feature extraction and svm for acoustic scene classification. In: 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. pp. 1–4. IEEE (2013)

[28] Ghalamchi, B., Jia, Z., Mueller, M.W.: Real-time vibration-based propeller fault diagnosis for multicopters. IEEE/ASME Transactions on Mechatronics 25(1), 395–405 (2019)

[29] Gomez, M.S., Koschlik, A.K., Arts, E., Raddatz, F.: Non-destructive evaluation of the condition of a uav's propellers by means of acoustics. In: NDE 4.0, Predictive Maintenance, and Communication and Energy Systems in a Globally Networked World. vol. 12049, pp. 22–30. SPIE (2022)

[30] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)

[31] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems 27 (2014)

[32] Gourisaria, M.K., Agrawal, R., Sahni, M., Singh, P.K.: Comparative analysis of audio classification with mfcc and stft features using machine learning techniques. Discover Internet of Things 4(1), 1 (2024)

[33] Grosse, R., Raina, R., Kwong, H., Ng, A.Y.: Shift-invariance sparse coding for audio classification. arXiv preprint arXiv:1206.5241 (2012)

[34] Harradi, R., Heller, A., Hardt, W.: Decentralized uav hangar: A study for water rescue missions. In: 2024 International Symposium on Computer Science and Educational Technology (ISCSET). pp. 1–4. IEEE (2024)

[35] Harradi, R., Heller, A., Roth, J., Hardt, W.: Mavlink uav hangar communication based on a cloud architecture. In: 2024 International Symposium ELMAR. pp. 301–305. IEEE (2024)

[36] Harras, M.S., Saleh, S., Battseren, B., Hardt, W.: Vision-based propeller damage inspection using machine learning. Embedded Selforganising Systems 10(7), 43–47 (2023)

[37] Hassanalian, M., Abdelkefi, A.: Classifications, applications, and design challenges of drones: A review. Progress in Aerospace sciences 91, 99–131 (2017)

[38] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, NY, 2nd edn. (2009)

[39] Heller, A., Harradi, R., Hardt, W.: Hangar system for unmanned aerial vehicle autonomous missions. In: 2024 International Symposium ELMAR. pp. 291–294. IEEE (2024)

[40] Iannace, G., Ciaburro, G., Trematerra, A.: Fault diagnosis for uav blades using artificial neural network. Robotics 8(3), 59 (2019)

[41] Iannace, G., Ciaburro, G., Trematerra, A.: Acoustical unmanned aerial vehicle detection in indoor scenarios using logistic regression model. Building Acoustics 28(1), 77–96 (2021)

[42] de Jesus Rangel-Magdaleno, J., Ureña-Ureña, J., Hernández, A., Perez-Rubio, C.: Detection of unbalanced blade on uav by means of audio signal. In: 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). pp. 1–5. IEEE (2018)

[43] Keane, J.F., Carr, S.S.: A brief history of early unmanned aircraft. Johns Hopkins APL Technical Digest 32(3), 558–571 (2013)

[44] Khan, O., Shahini, G., Hardt, W.: Analysis of machine learning approach for the model in swc mapping in automotive systems. Embedded Selforganising Systems 7(1), 16–19 (2020)

[45] Kołodziejczak, M., Puchalski, R., Bondyra, A., Sladic, S., Giernacki, W.: Toward lightweight acoustic fault detection and identification of uav rotors. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 990–997. IEEE (2023)

[46] Kong, Q., Xu, Y., Sobieraj, I., Wang, W., Plumbley, M.D.: Sound event detection and time–frequency segmentation from weakly labelled data. IEEE/ACM Transactions on Audio, Speech, and Language Processing 27(4), 777–787 (2019)

[47] Krishnamurthi, R., Gopinathan, D., Kumar, A.: Using wavelet transformation for acoustic signal processing in heavy vehicle detection and classification. In: Autonomous and Connected Heavy Vehicle Technology, pp. 199–209. Elsevier (2022)

[48] Lacoste, R.: Chapter 6 - the fast fourier transform from a to z. In: Lacoste, R. (ed.) Robert Lacoste's The Darker Side, pp. 79–92. Newnes, Boston (2010), https://www.sciencedirect.com/science/article/pii/B978185617762700006X

[49] Le, T.T., Fu, W., Moore, J.H.: Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics 36(1), 250–256 (2020)

[50] Lee, J.y., Lee, W.t., Ko, S.h., Oh, H.s.: Fault classification and diagnosis of uav motor based on estimated nonlinear parameter of steady-state model. Int. J. Mech. Eng. Robot. Res 10(1), 22–31 (2020)

[51] Lim, W., Suh, S., Jeong, Y.: Weakly labeled semi-supervised sound event detection using crnn with inception module. In: DCASE. pp. 74–77 (2018)

[52] Liu, J., Lin, Q., Liu, G., Liu, D., Liu, J.: Airworthiness technology: The key for the development and application of civilian unmanned aircraft systems. In: International Conference on Autonomous Unmanned Systems. pp. 80–90. Springer (2023)

[53] Liu, M., Li, L., Yan, F.: Methods of fault diagnosis and prediction. In: Intelligent Predictive Maintenance, pp. 47–95. Springer (2024)

[54] Liu, R.l., Zhang, Z.j., Jiao, Y.f., Yang, C.h., Zhang, W.j.: Study on flight performance of propeller-driven uav. International Journal of Aerospace Engineering 2019(1), 6282451 (2019)

[55] Liu, W., Chen, Z., Zheng, M.: An audio-based fault diagnosis method for quadrotors using convolutional neural network and transfer learning. In: 2020 American Control Conference (ACC). pp. 1367–1372. IEEE (2020)

[56] Liu, W., Liu, C., Sajedi, S., Su, H., Liang, X., Zheng, M.: An audio-based risky flight detection framework for quadrotors. IET Cyber-Systems and Robotics 6(1), e12105 (2024)

[57] Loizou, P.C.: Speech enhancement: theory and practice. CRC press (2007)

[58] McFee, B., Raffel, C., Liang, D., Ellis, D.P., McVicar, M., Battenberg, E., Nieto, O.: librosa: Audio and music signal analysis in python. SciPy 2015, 18–24 (2015)

[59] OpenCourseWare, M.: Students in class at mit. `https://www.flickr.com/photos/mitopencourseware/3042950125/in/photostream/` (November 2008), accessed: 2025-03-04

[60] Organization, W.H.: Hidden depths: the global investment case for drowning prevention. World Health Organization (2023)

[61] Palanisamy, R.P., Kulkarni, C.S., Corbetta, M., Banerjee, P.: Fault detection and performance monitoring of propellers in electric uav. In: 2022 IEEE Aerospace Conference (AERO). pp. 1–6. IEEE (2022)

[62] Parker, M.: Chapter 3 - sampling, aliasing, and quantization. In: Parker, M. (ed.) Digital Signal Processing 101 (Second Edition), pp. 21–30. Newnes, second edition edn. (2017), `https://www.sciencedirect.com/science/article/pii/B9780128114537000032`

[63] Pechan, T., Sescu, A.: Experimental study of noise emitted by propeller's surface imperfections. Applied Acoustics 92, 12–17 (2015)

[64] Podsdkowski, M., Konopiński, R., Lipian, M.: Acoustic stall detection of variable pitch propeller for unmanned aerial vehicles. Journal of Intelligent & Robotic Systems 109(3), 70 (2023)

[65] Poorghasem, S., Bao, Y.: Review of robot-based automated measurement of vibration for civil engineering structures. Measurement 207, 112382 (2023)

[66] Pose, C., Giribet, J., Torre, G.: Propeller damage detection, classification and estimation in multirotor vehicles. arXiv preprint arXiv:2410.05447 (2024)

[67] Puchalski, R., Giernacki, W.: Uav fault detection methods, state-of-the-art. Drones 6(11), 330 (2022)

[68] Ray, D.K., Roy, T., Chattopadhyay, S.: Skewness scanning for diagnosis of a small inter-turn fault in quadcopter's motor based on motor current signature analysis. IEEE Sensors Journal 21(5), 6952–6961 (2020)

[69] RescueFly: Projektbeschreibung. `https://rescuefly.org/projektbeschreibung/` (2024), [Online; accessed 31-October-2024]

[70] ResearchGate: Sampling of audio signal. `https://www.researchgate.net/figure/Sampling-of-audio-signal_fig3_266488076` (2014), accessed: 2025-03-04

[71] Restas, A.: Drone applications for supporting disaster management. World Journal of Engineering and Technology 3(3), 316–321 (2015)

[72] Saleh, S., Manoharan, S., Nine, J., Hardt, W.: Towards robust perception depth information for collision avoidance. In: 2020 IEEE Congreso Bienal de Argentina (ARGENCON). pp. 1–4. IEEE (2020)

[73] Sanket Doshi: Audio signal. `https://medium.com/towards-data-science/extract-features-of-music-75a3f9bc265d`, accessed: 2018-12-30

[74] Sarhan, A., Qin, S.: Autonomous intelligent flight control of fixed-wing uav based on adaptive neuro-fuzzy inference system. International Journal of Research in Engineering and Technology 5(9), 92–100 (2016)

[75] Seguin, C., Blaquière, G., Loundou, A., Michelet, P., Markarian, T.: Unmanned aerial vehicles (drones) to prevent drowning. Resuscitation 127, 63–67 (2018)

[76] Semke, W.H., Zahui, D.K., Schwalb, J.: The vibration and acoustic effects of prop design and unbalance on small unmanned aircraft. In: Sensors and

Instrumentation, Aircraft/Aerospace, Energy Harvesting & Dynamic Environments Testing, Volume 7: Proceedings of the 38th IMAC, A Conference and Exposition on Structural Dynamics 2020. pp. 9–16. Springer (2021)

[77] Skiadopoulos, A., Stergiou, N.: Chapter 5 - power spectrum and filtering. In: Stergiou, N. (ed.) Biomechanics and Gait Analysis, pp. 99–148. Academic Press (2020), https://www.sciencedirect.com/science/article/pii/B9780128133729000051

[78] Škvorc, P., Kozmar, H.: Wind energy harnessing on tall buildings in urban environments. Renewable and Sustainable Energy Reviews 152, 111662 (2021)

[79] Soria Gomez, M., Koschlik, A.K., Arts, E., Raddatz, F., Wende, G.: Acoustic non-destructive testing of uas´ s propellers during predeparture and postflight checks. In: Proceedings of the 13th European Conference on Non-Destructive Testing 2023 (2023)

[80] Steinhoff, L., Koschlik, A.K., Arts, E., Soria-Gomez, M., Raddatz, F., Kunz, V.D.: Development of an acoustic fault diagnosis system for uav propeller blades. CEAS Aeronautical Journal 15(4), 881–893 (2024)

[81] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27. pp. 270–279. Springer (2018)

[82] Tan, L., Jiang, J.: Chapter 1 - introduction to digital signal processing. In: Tan, L., Jiang, J. (eds.) Digital Signal Processing (Third Edition), pp. 1–12. Academic Press, third edition edn. (2019), https://www.sciencedirect.com/science/article/pii/B9780128150719000014

[83] Tong, J., Zhang, W., Liao, F., Li, C., Zhang, Y.: Machine learning for uav propeller fault detection based on a hybrid data generation model. arXiv preprint arXiv:2302.01556 (2023)

[84] Toylan, H., Türkeş, E., Çağlarer, E.: Real-time control of mobile robot using hmm-based speech recognition system. Anadolu University Journal of Science and Technology A-Applied Sciences and Engineering 18(5), 897–907 (2017)

[85] Tudevdagva, U., Battseren, B., Hardt, W., Blokzyl, S., Lippmann, M.: Unmanned aerial vehicle-based fully automated inspection system for high voltage transmission line. In: Proceedings on the 12th International Forum on Strategic Technology IEEE conference, IFOST 2017. pp. 300–305 (2017)

[86] Utebayeva, D., Almagambetov, A., Alduraibi, M., Temirgaliyev, Y., Ilipbayeva, L., Marxuly, S.: Multi-label uav sound classification using stacked bidirectional lstm. In: 2020 fourth IEEE international conference on robotic computing (IRC). pp. 453–458. IEEE (2020)

[87] Von Beesten, J., et al.: Rescuefly-einsatz von dezentral station-ierten drohnen ("unmanned aircraft systems. UAS) zur Unterstützung bei der Wasserrettung in schwer zugänglichen und weitflächigen Gebieten," Brandenburgisches Institut für Gesellschaft und Sicherheit gGmbH, Potsdam, Germany, Rep. 11 (2024)

[88] Whiteside, S., Zawodny, N., Fei, X., Pettingill, N.A., Patterson, M.D., Rothhaar, P.: An exploration of the performance and acoustic characteristics of uav-scale stacked rotor configurations. In: AIAA Scitech 2019 Forum. p. 1071 (2019)

[89] Yaman, O., Tuncer, T., Tasar, B.: Des-pat: A novel des pattern-based propeller recognition method using underwater acoustical sounds. Applied Acoustics 175, 107859 (2021)

[90] Yaman, O., Yol, F., Altinors, A.: A fault detection method based on embedded feature extraction and svm classification for uav motors. Microprocessors and Microsystems 94, 104683 (2022)

[91] Yang, B., Matson, E.T., Smith, A.H., Dietz, J.E., Gallagher, J.C.: Uav detection system with multiple acoustic nodes using machine learning models. In: 2019 Third IEEE international conference on robotic computing (IRC). pp. 493–498. IEEE (2019)

[92] Yasuda, Y.D., Cappabianco, F.A., Martins, L.E.G., Gripp, J.A.: Aircraft visual inspection: A systematic literature review. Computers in Industry 141, 103695 (2022)

[93] Yol, F., Altınors, A., Yaman, O.: A sound based method for fault classification with support vector machines in uav motors. International Journal of Data Science and Applications 4(1), 5–10 (2021)

[94] Yong, L.Z., Nugroho, H.: Acoustic anomaly detection of mechanical failure: Time-distributed cnn-rnn deep learning models. In: Control, Instrumentation and Mechatronics: Theory and Practice, pp. 662–672. Springer (2022)

[95] Zhang, B., Song, Z., Zhao, F., Liu, C.: Overview of propulsion systems for unmanned aerial vehicles. Energies 15(2), 455 (2022)

[96] Zhang, Q.Y., Hu, W.J., Qiao, S.B., Huang, Y.B., et al.: Speech perceptual hashing authentication algorithm based on spectral subtraction and energy to entropy ratio. Int. J. Netw. Secur. 19(5), 752–760 (2017)

[97] Zhao, H., Yang, W., Zhu, H.: Unmanned aerial vehicles rescue system design and traffic model planning. Applied Sciences 11(21), 10481 (2021)

[98] Zhu, Q., Zhou, R., Zhang, J.: Connectivity maintenance based on multiple relay uavs selection scheme in cooperative surveillance. Applied Sciences 7(1), 8 (2016)

[99] Zuo, L., Yao, L., Kang, Y.: Uio based sensor fault diagnosis and compensation for quadrotor uav. In: 2020 Chinese Control And Decision Conference (CCDC). pp. 4052–4057. IEEE (2020)

References of Computer Engineering's Professorship

[7] Battseren, B., Tudevdagva, U., Hardt, W., & Bilegt, D. (2024). Development of an Adaptive MAV Platform for Autonomous Inspection of High Voltage Power Lines. In *NEIS - Conf. Sustain. Energy Supply Energy Storage Syst.*

[34] Harradi, R., Heller, A., & Hardt, W. (2024). Decentralized UAV hangar: A study for water rescue missions. In *2024 International Symposium on Computer Science and Educational Technology (ISCSET)* (pp. 1–4). IEEE.

[35] Harradi, R., Heller, A., Roth, J., & Hardt, W. (2024). MAVLink UAV hangar communication based on a cloud architecture. In *2024 International Symposium ELMAR* (pp. 301–305). IEEE.

[36] Harras, M. S., Saleh, S., Battseren, B., & Hardt, W. (2023). Vision-based Propeller Damage Inspection Using Machine Learning. *Embedded Selforganising Systems*, 10(7), 43–47.

[39] Heller, A., Harradi, R., & Hardt, W. (2024). Hangar system for unmanned aerial vehicle autonomous missions. In *2024 International Symposium ELMAR* (pp. 291–294). IEEE.

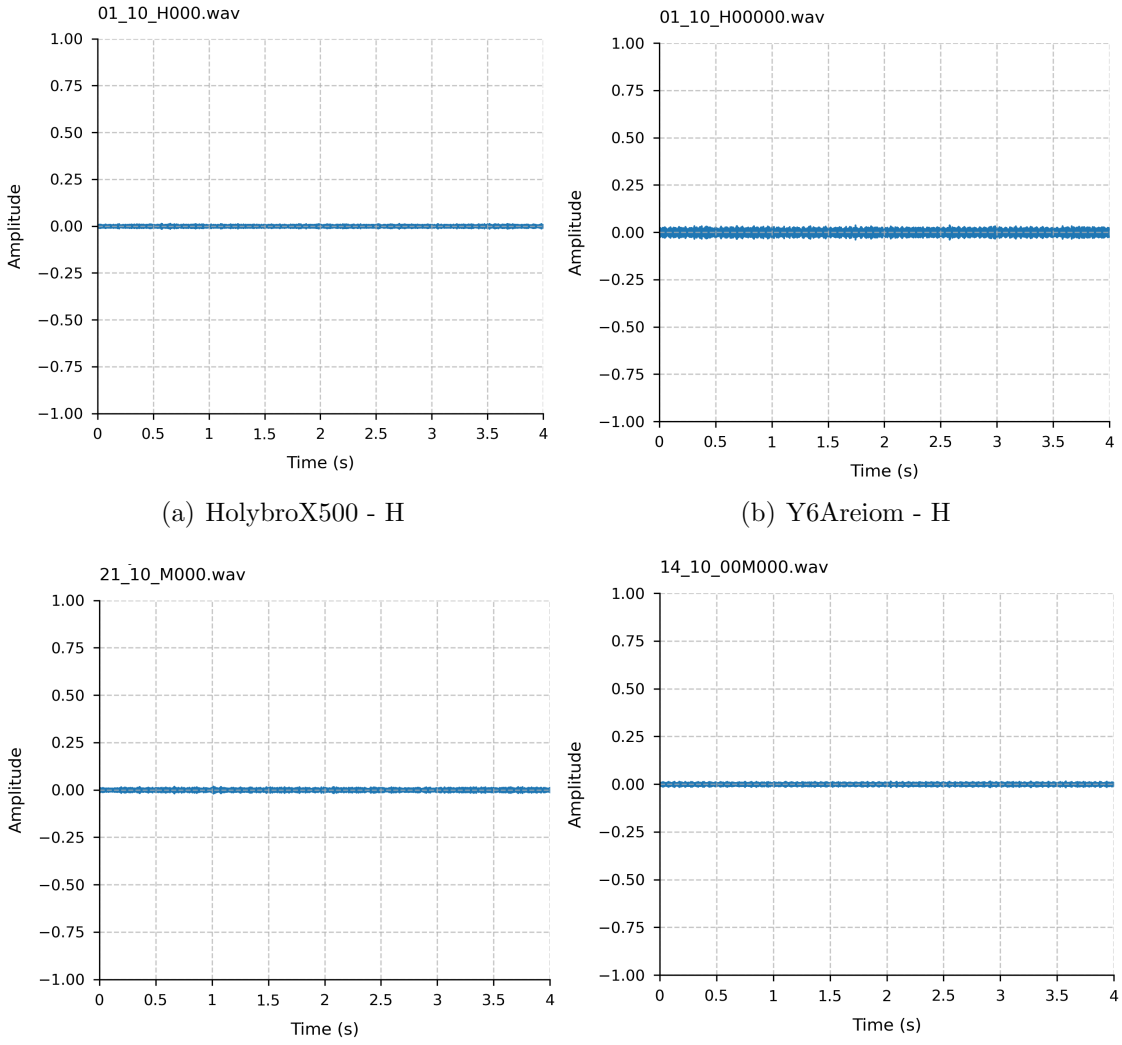
[44] Khan, O., Shahini, G., & Hardt, W. (2020). Analysis of Machine Learning Approach for the model in SWC Mapping in Automotive Systems. *Embedded Selforganising Systems*, 7(1), 16–19.

[72] Saleh, S., Manoharan, S., Nine, J., & Hardt, W. (2020). Towards robust perception depth information for collision avoidance. In *2020 IEEE Congreso Bienal de Argentina (ARGENCON)* (pp. 1–4). IEEE.

[85] Tudevdagva, U., Battseren, B., Hardt, W., Blokzyl, S., & Lippmann, M. (2017). UAV Based Fully Automated Inspection System for High Voltage Transmission Lines. In *12th International Forum on Strategic Technology (IFOST)*, Ulsan, Korea, May 2017.

A. Supplementary Figures

A.1. Waveform Analysis



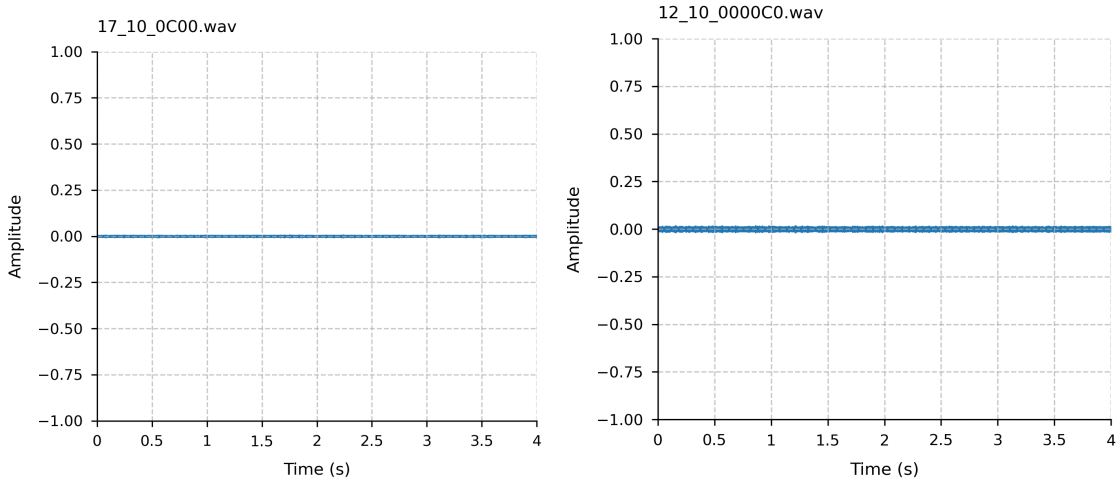
(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

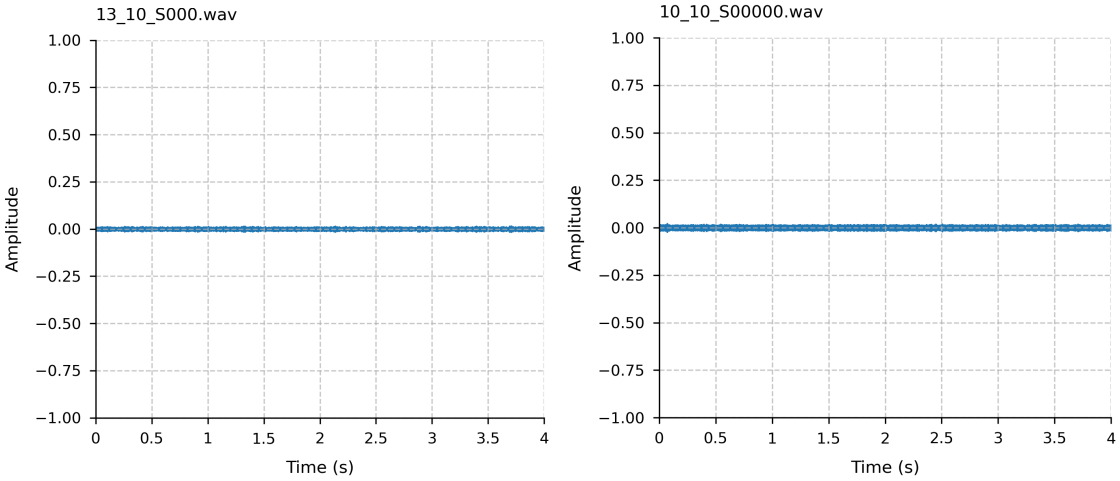
(d) Y6Areiom - M

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed



(e) HolybroX500 - C

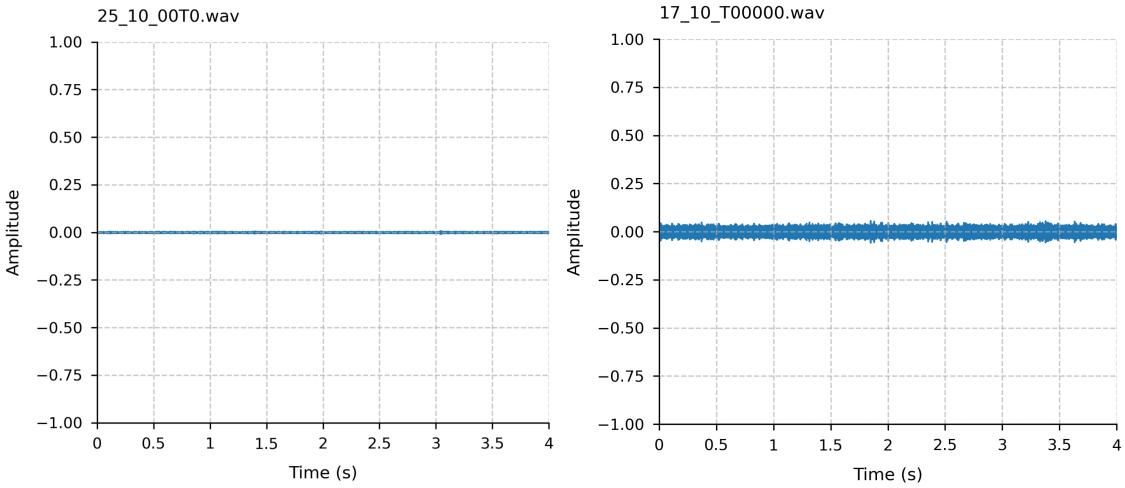
(f) Y6Areiom - C



(g) HolybroX500 - S

(h) Y6Areiom - S

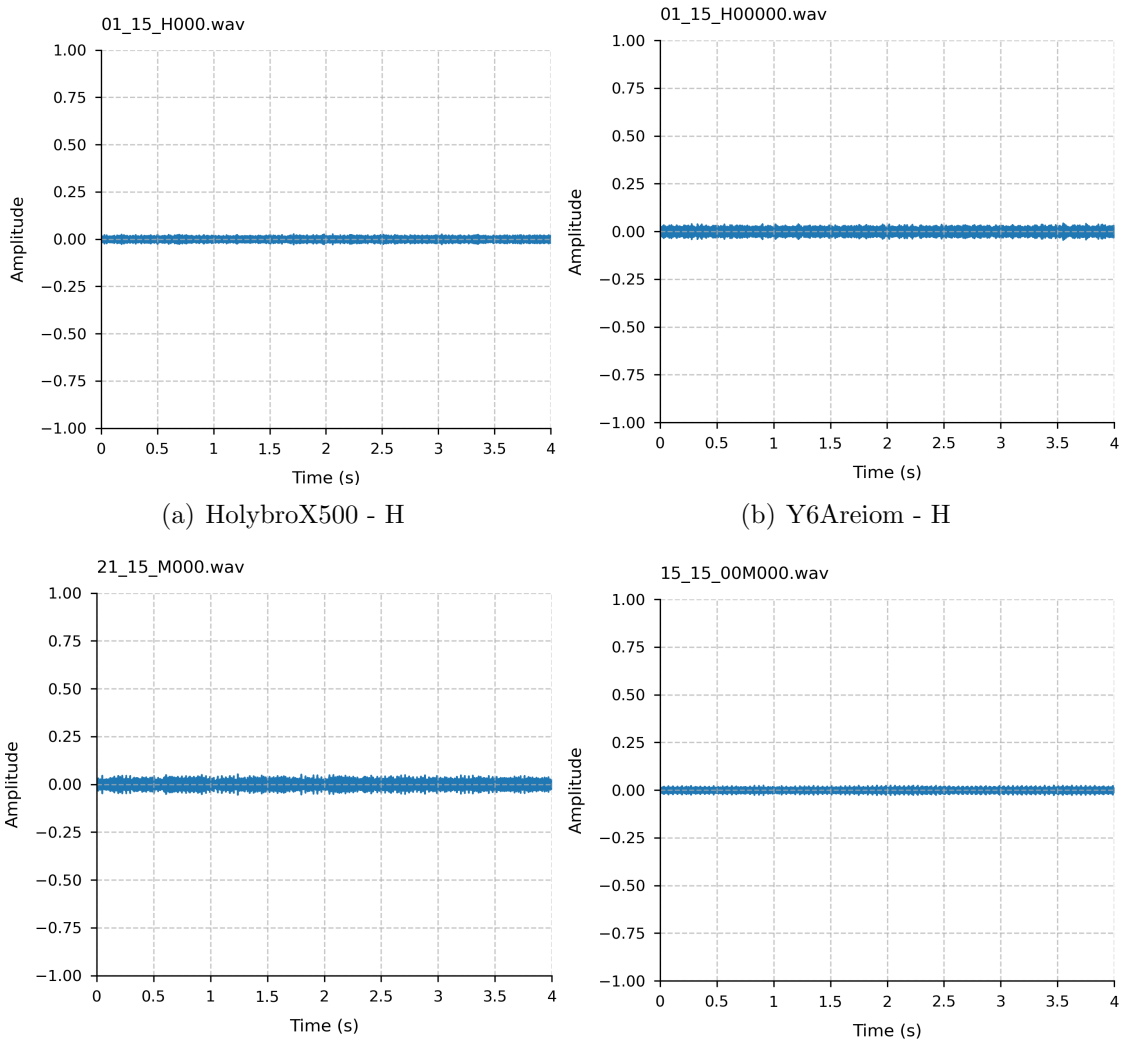
Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed (cont.)



(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed (cont.)



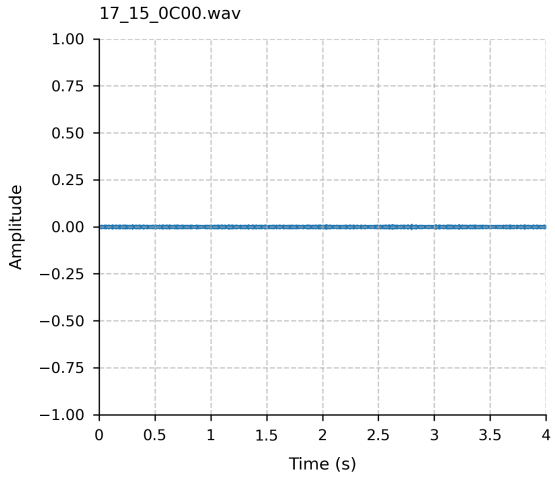
(a) HolybroX500 - H

(b) Y6Areiom - H

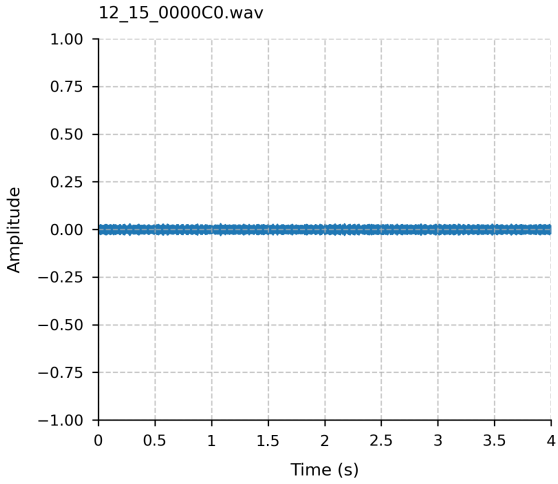
(c) HolybroX500 - M

(d) Y6Areiom - M

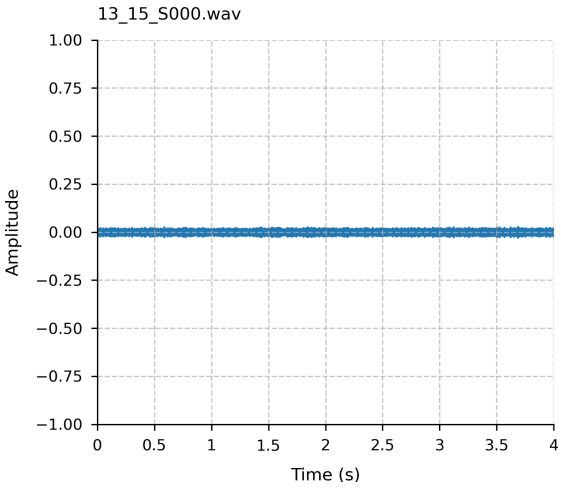
Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed



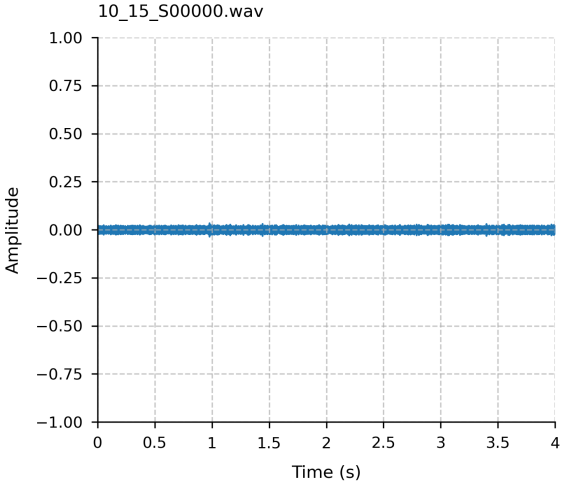
(e) HolybroX500 - C



(f) Y6Areiom - C

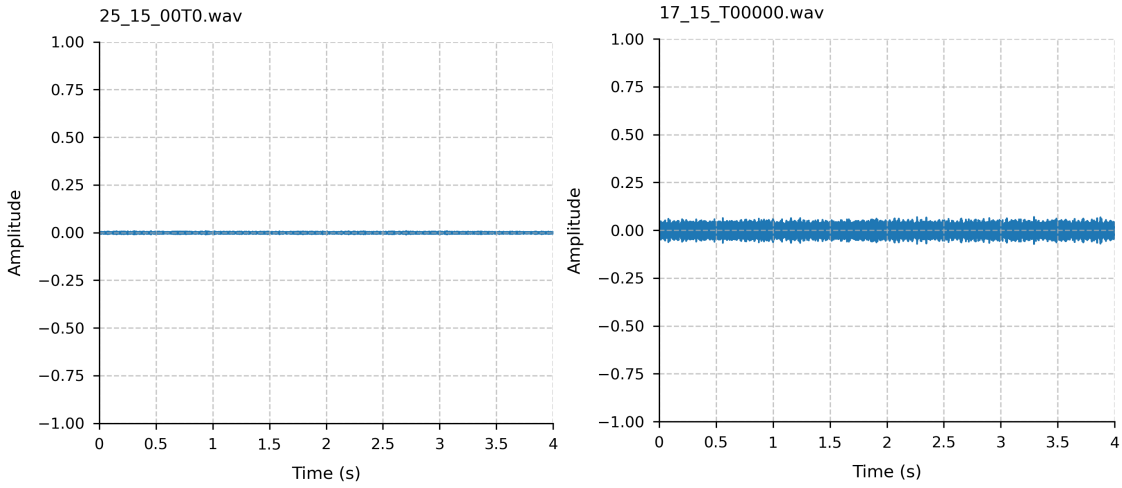


(g) HolybroX500 - S



(h) Y6Areiom - S

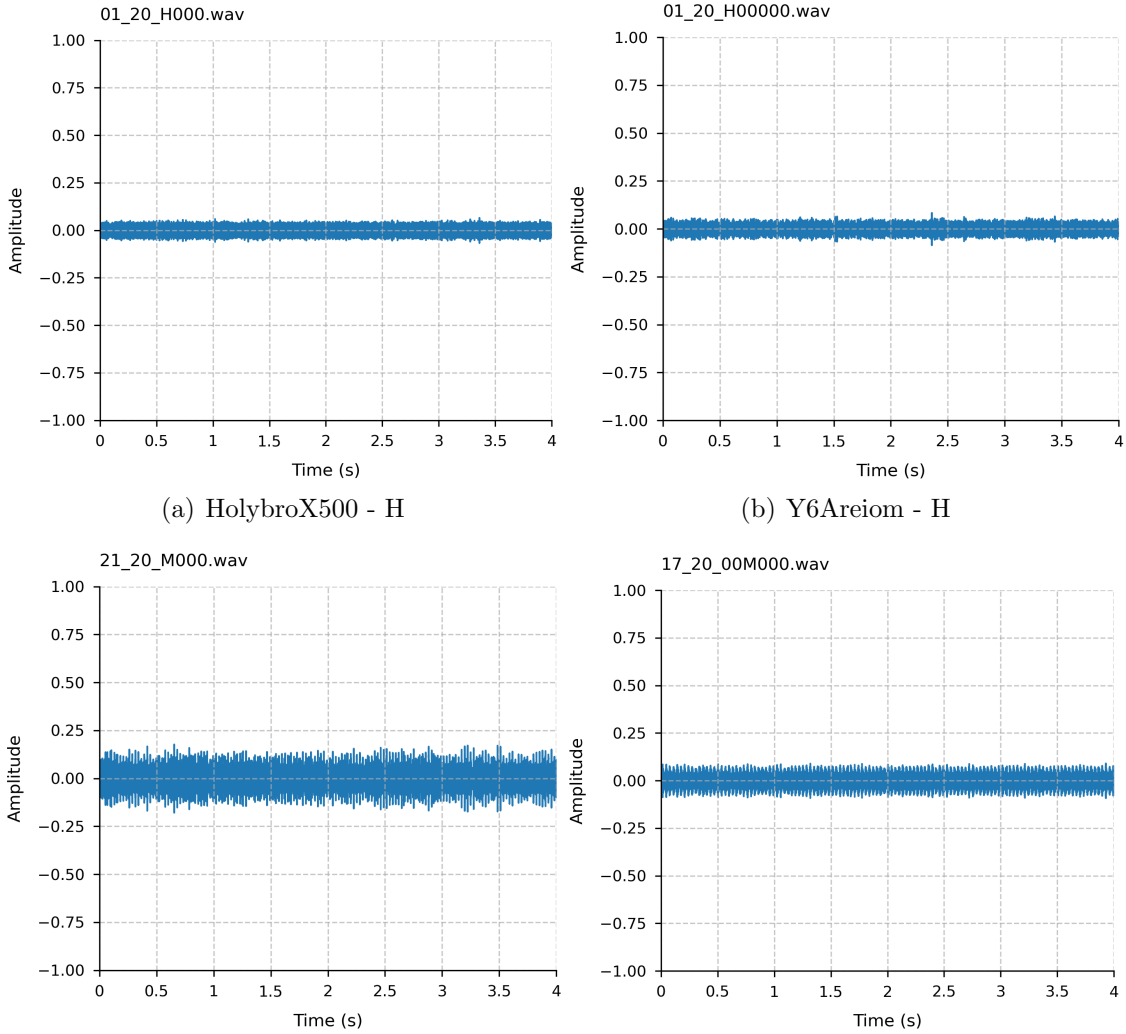
Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed (cont.)



(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed (cont.)



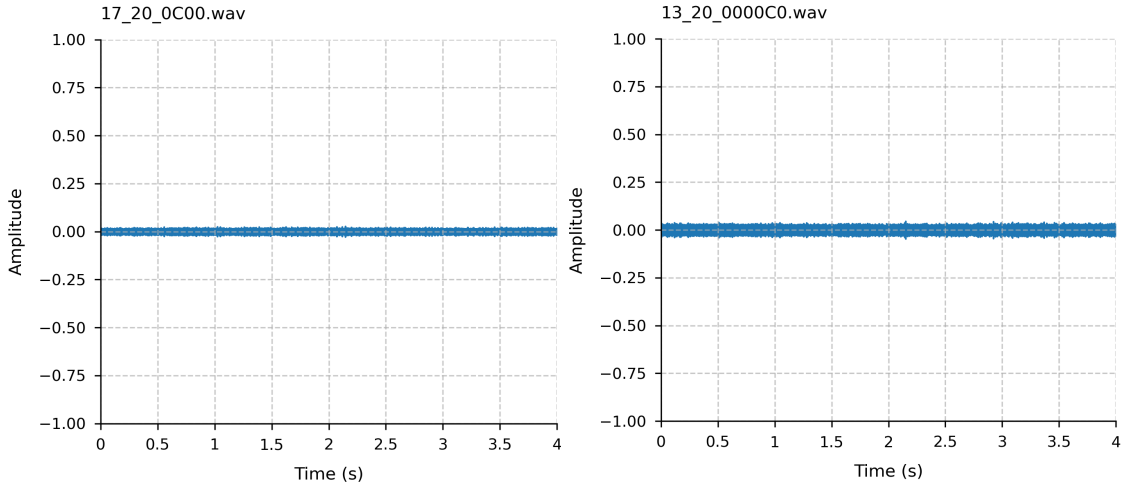
(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

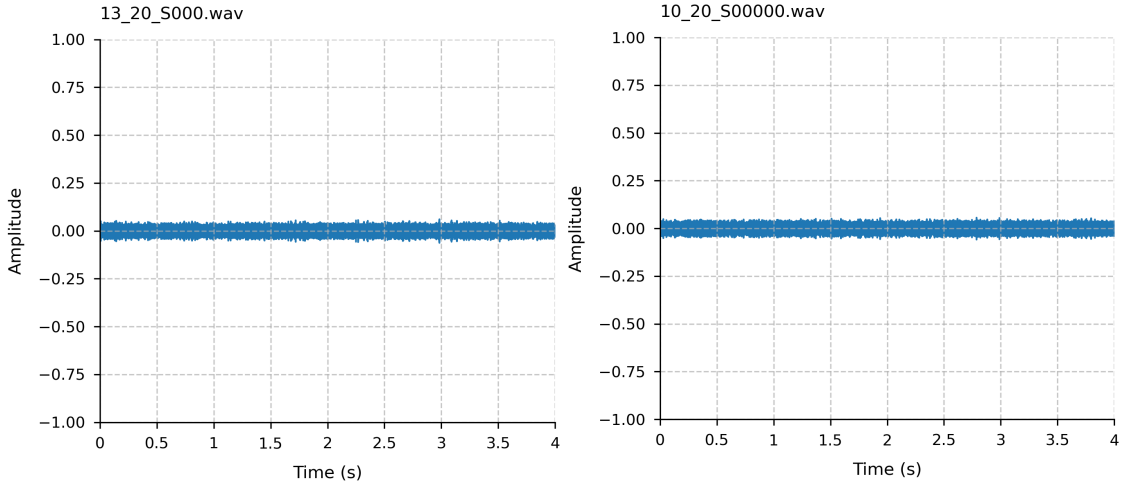
(d) Y6Areiom - M

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed



(e) HolybroX500 - C

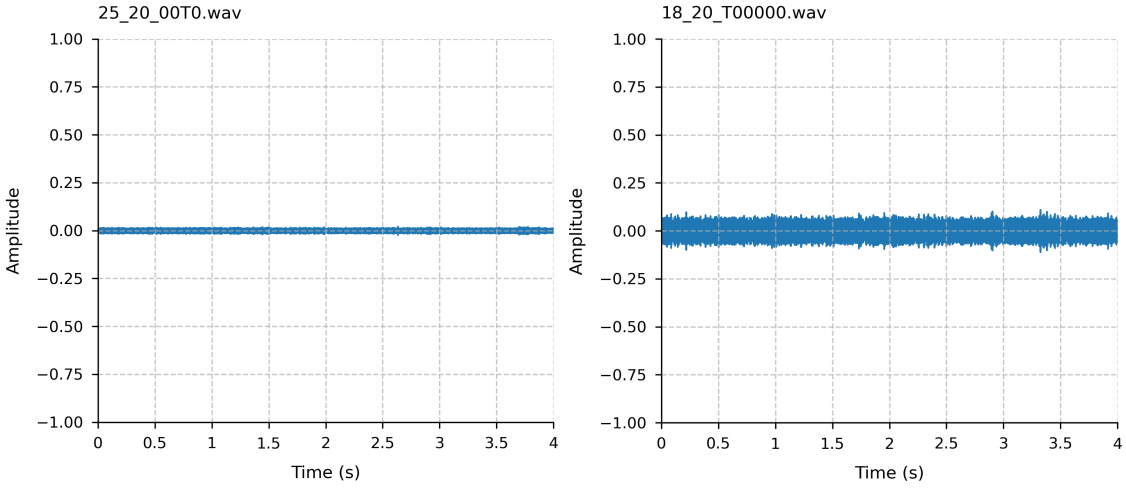
(f) Y6Areiom - C



(g) HolybroX500 - S

(h) Y6Areiom - S

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed (cont.)

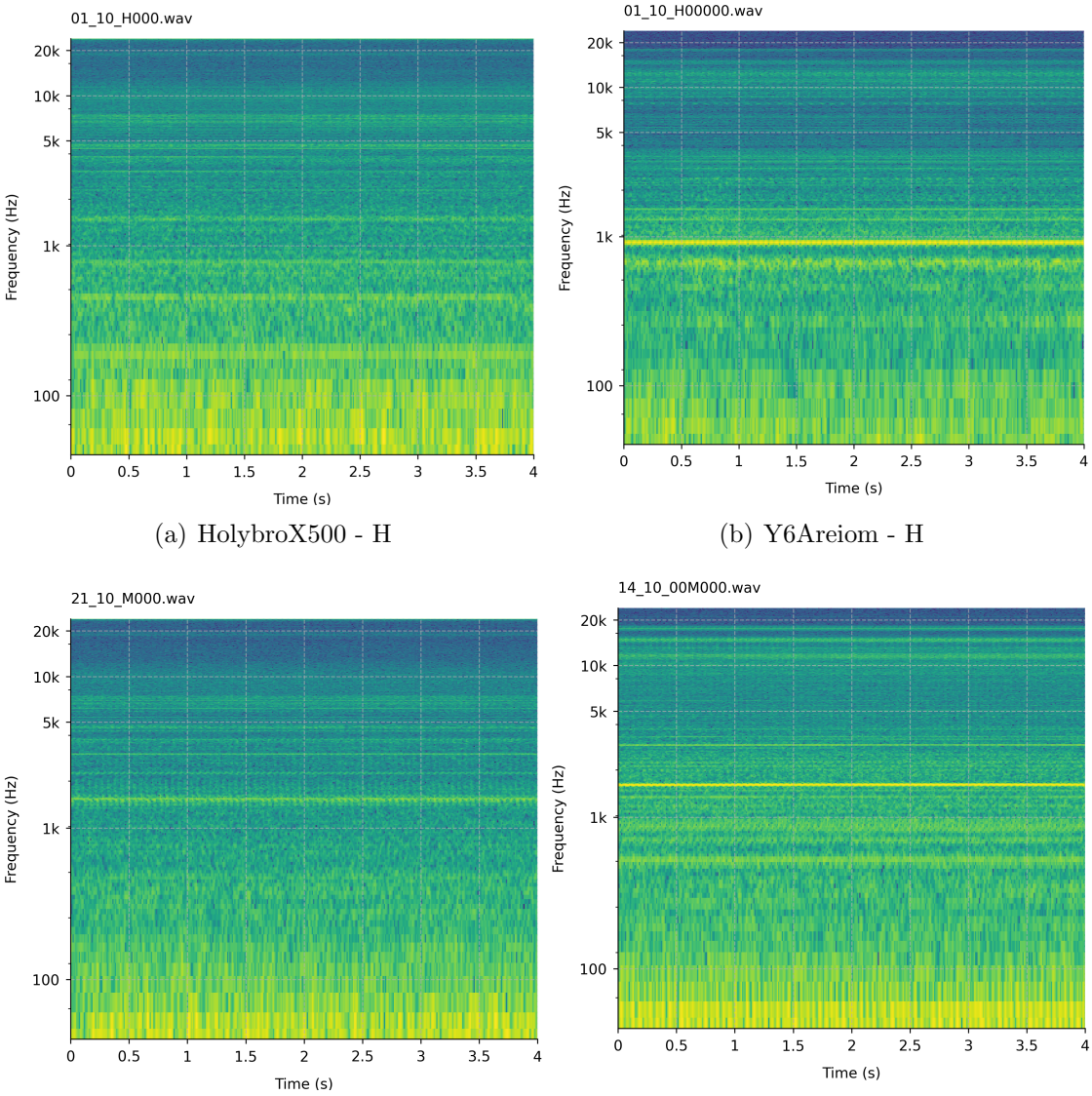


(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Waveform analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed (cont.)

A.2. Spectrogram Analysis



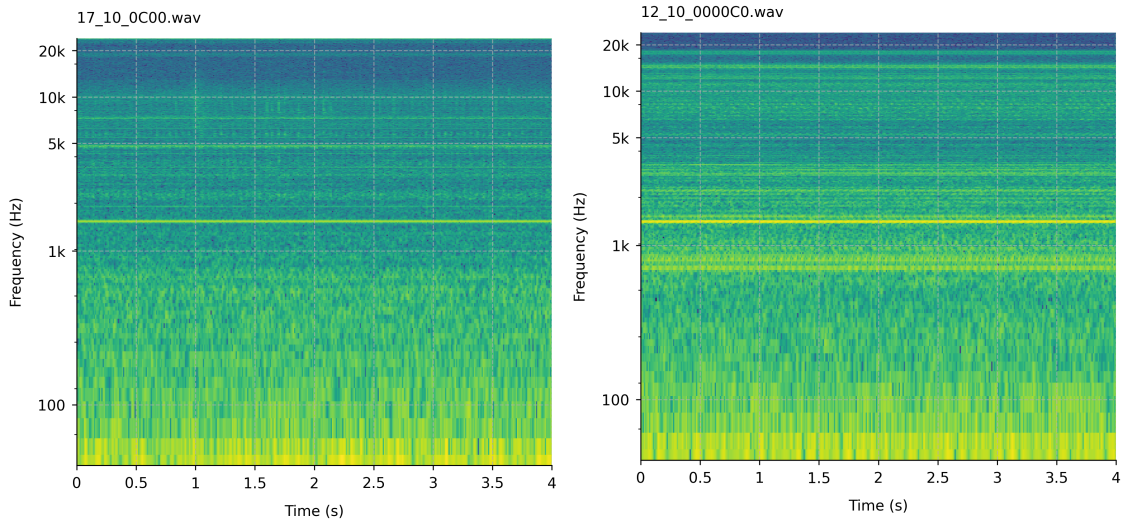
(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

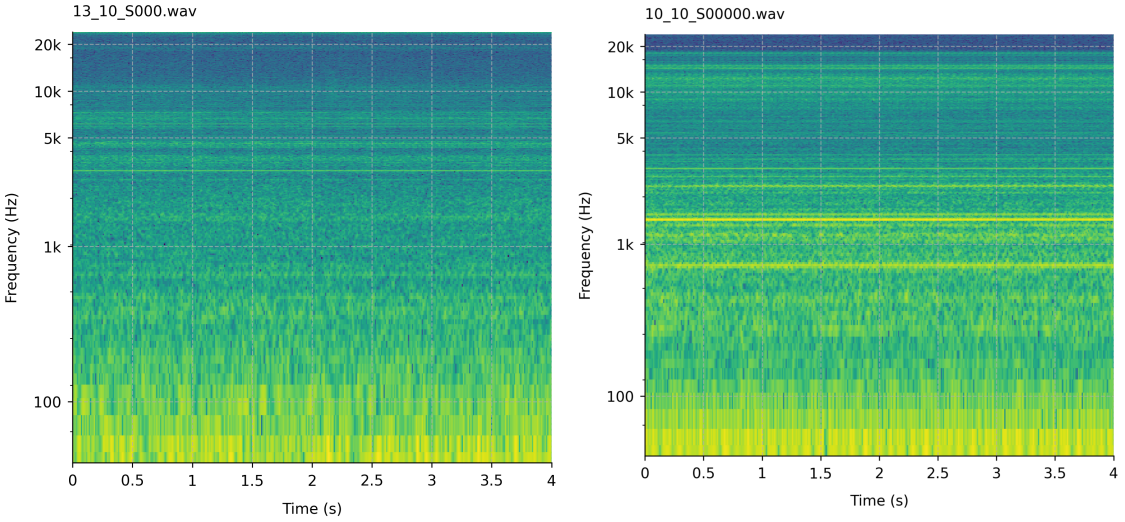
(d) Y6Areiom - M

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed



(e) HolybroX500 - C

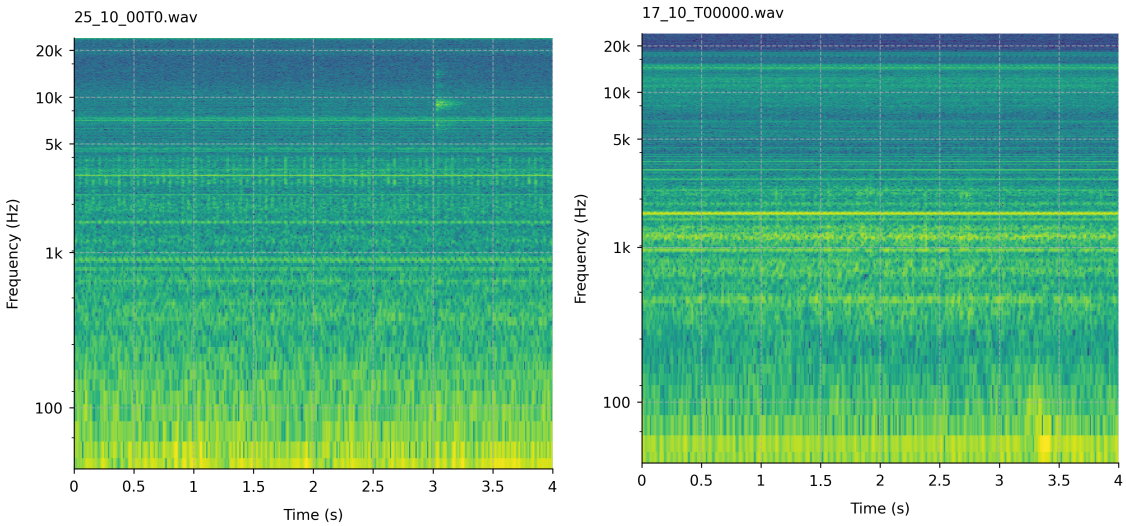
(f) Y6Areiom - C



(g) HolybroX500 - S

(h) Y6Areiom - S

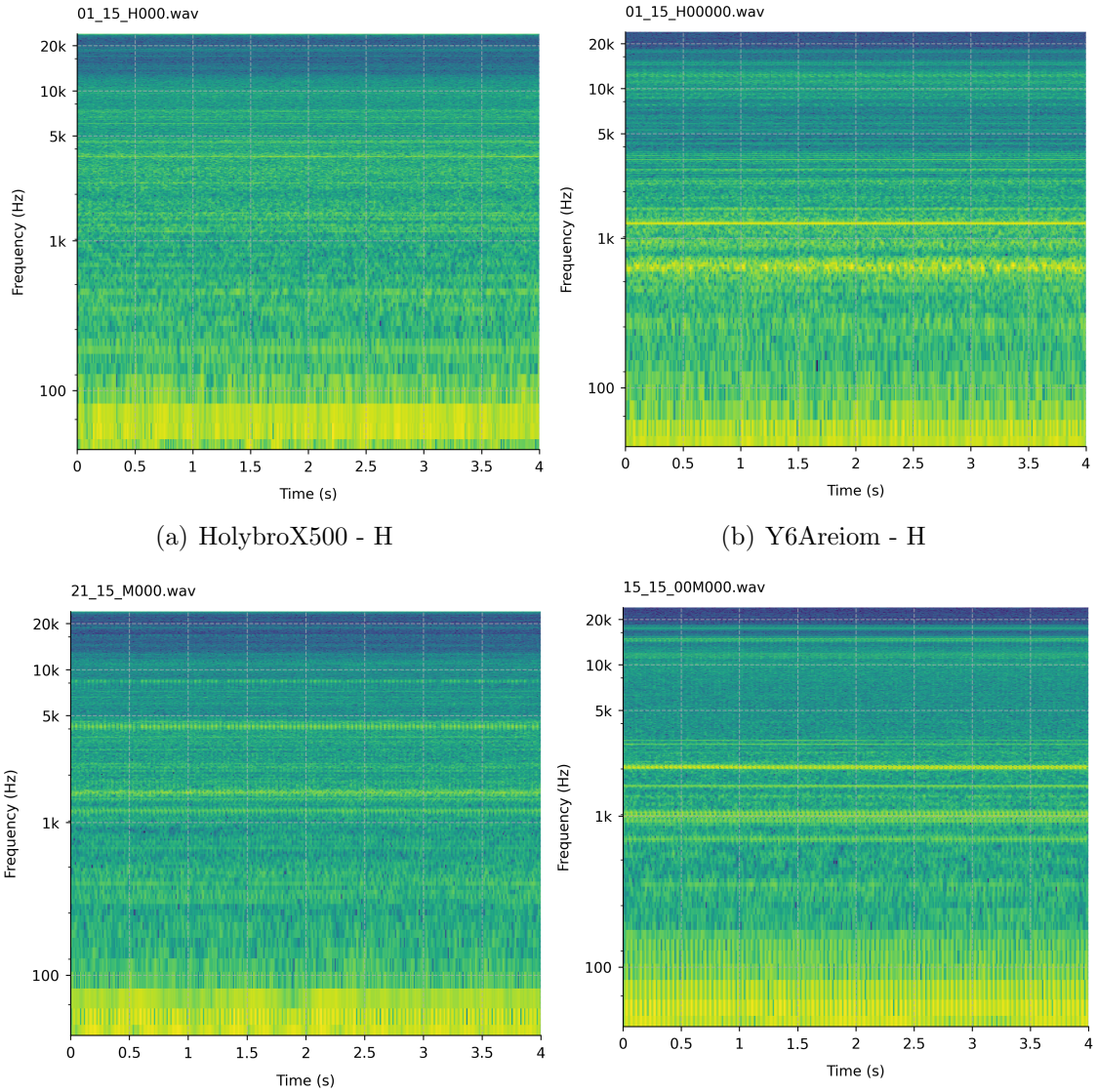
Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed (cont.)



(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 10% speed (cont.)



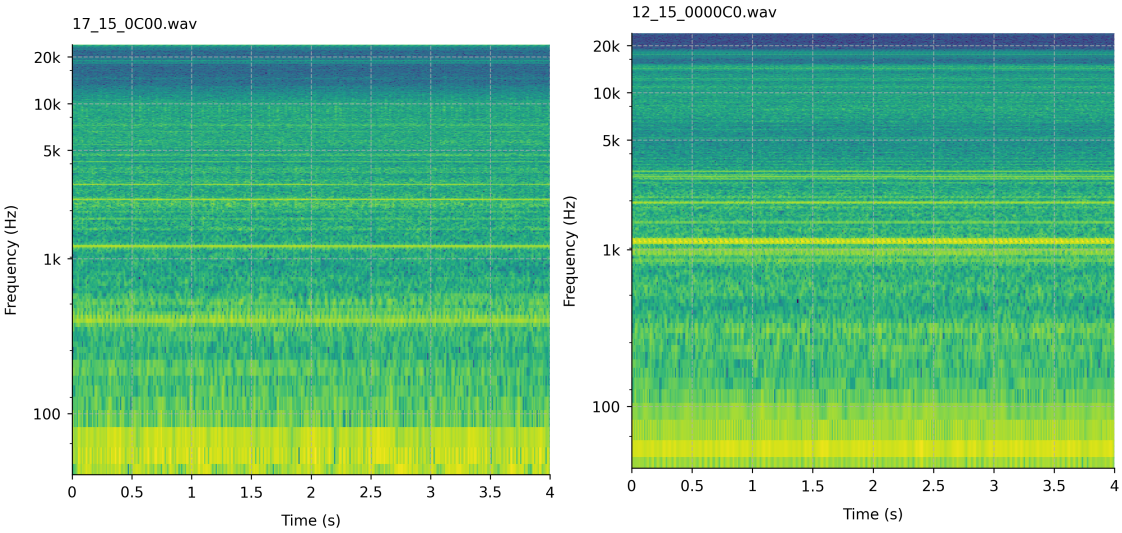
(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

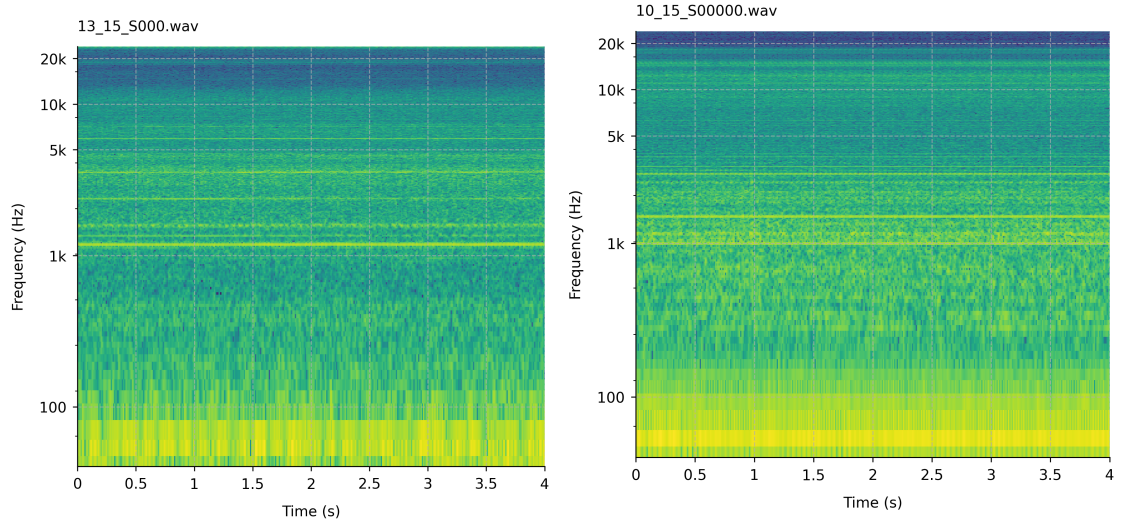
(d) Y6Areiom - M

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed



(e) HolybroX500 - C

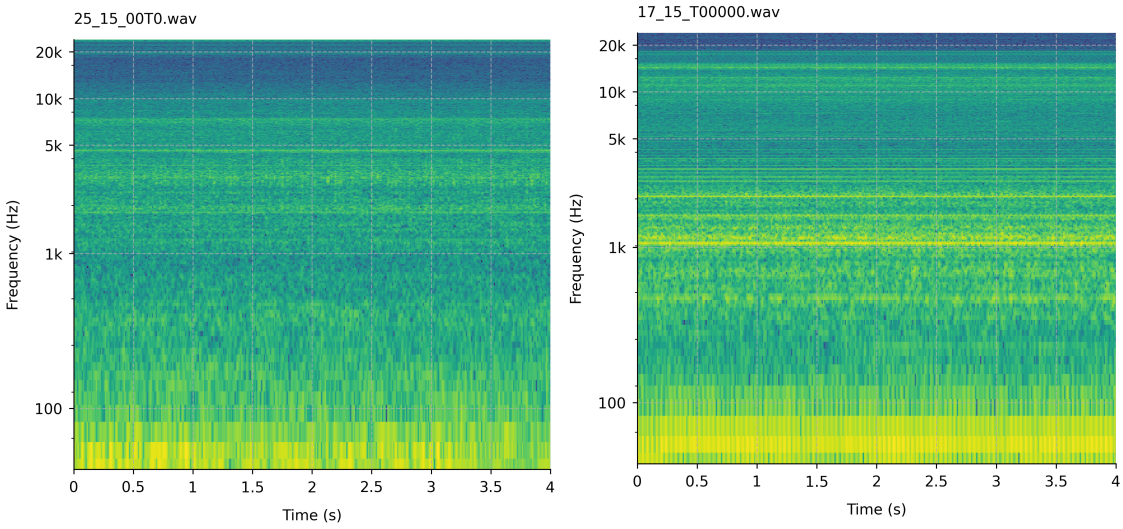
(f) Y6Areiom - C



(g) HolybroX500 - S

(h) Y6Areiom - S

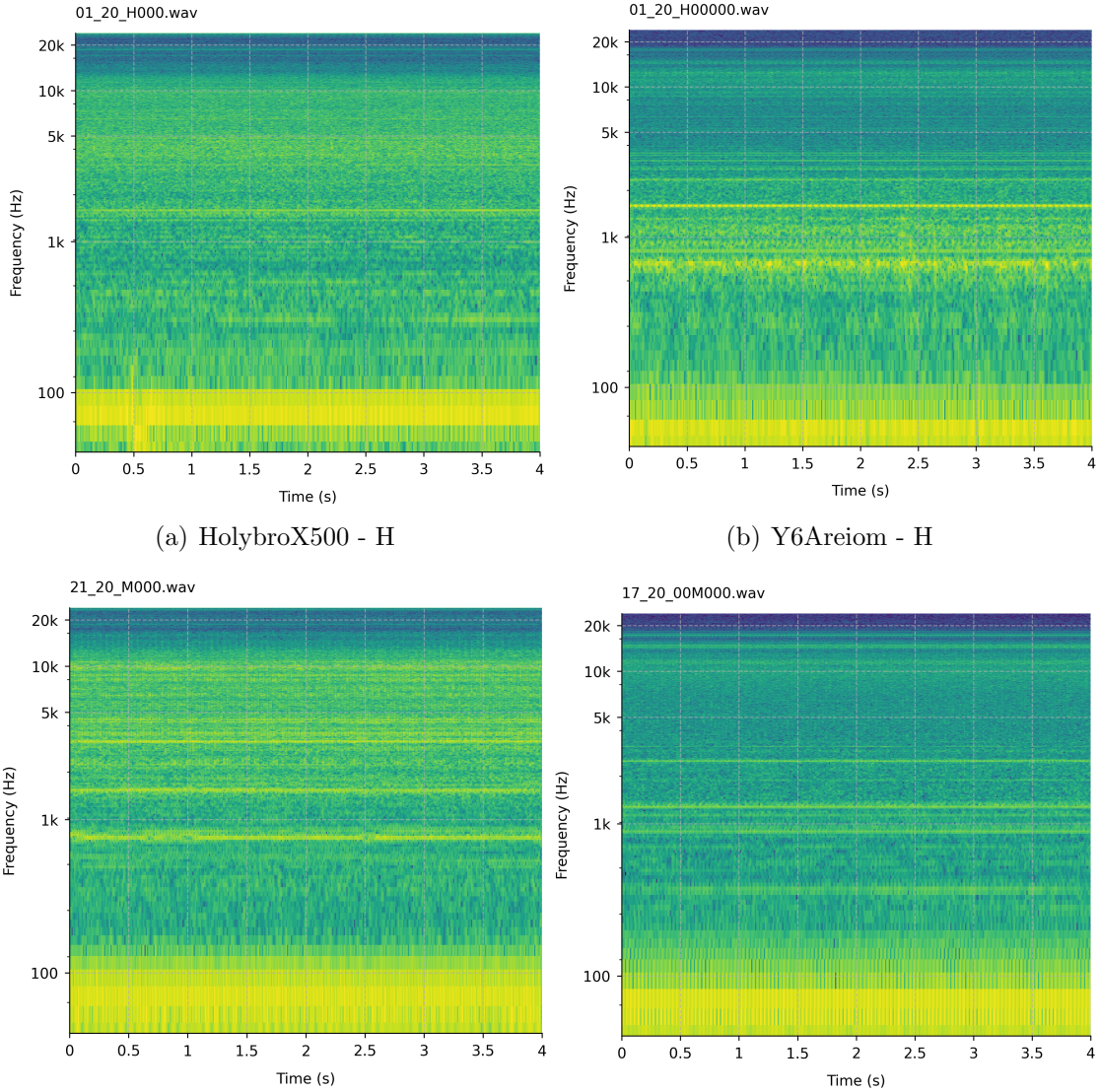
Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed (cont.)



(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 15% speed (cont.)



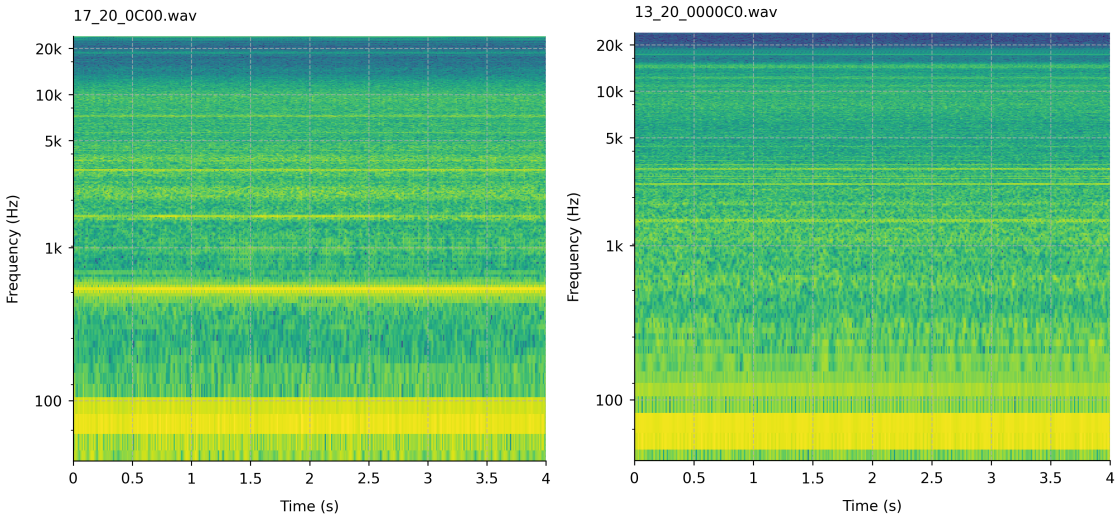
(a) HolybroX500 - H

(b) Y6Areiom - H

(c) HolybroX500 - M

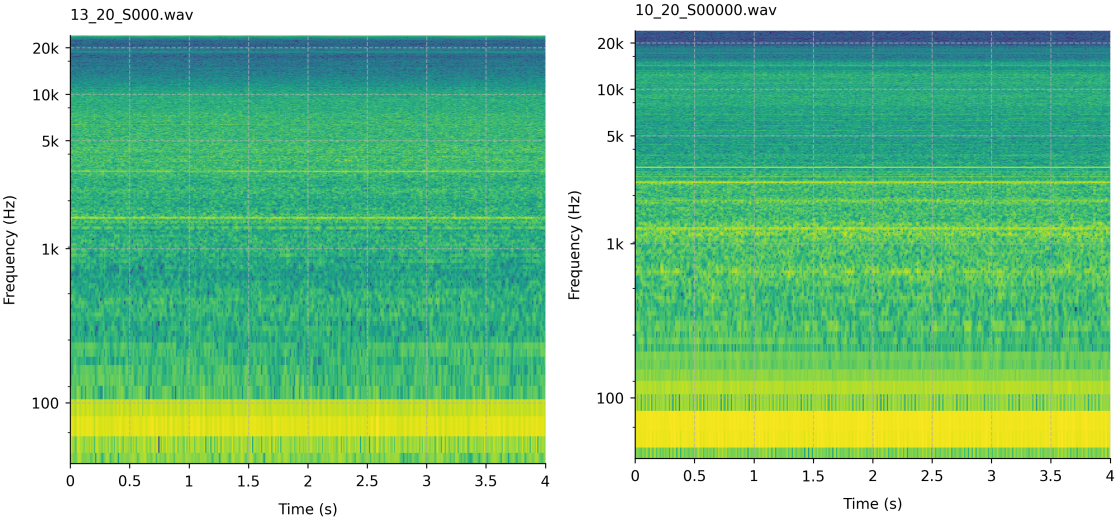
(d) Y6Areiom - M

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed



(e) HolybroX500 - C

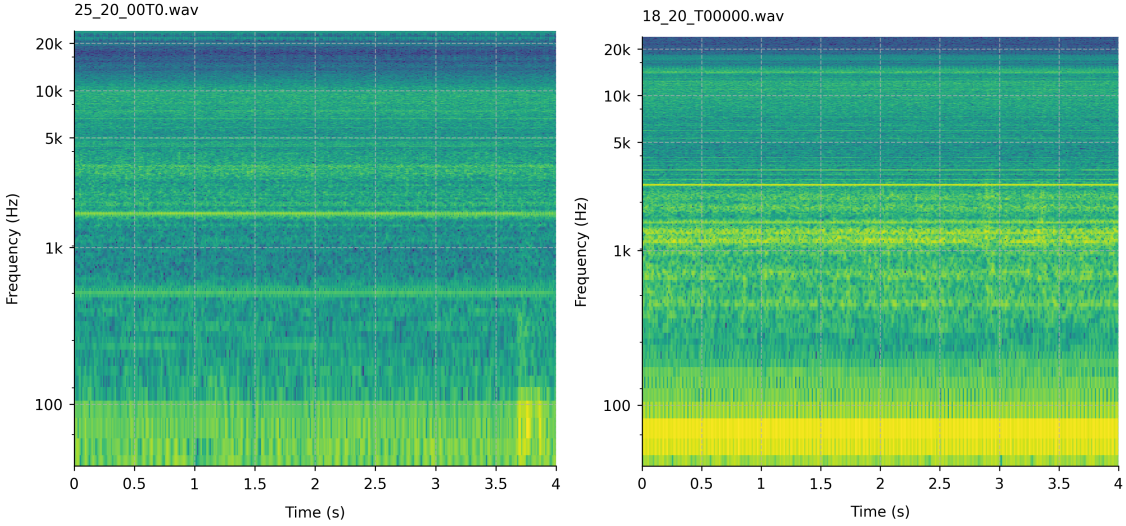
(f) Y6Areiom - C



(g) HolybroX500 - S

(h) Y6Areiom - S

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed (cont.)



(i) HolybroX500 - T

(j) Y6Areiom - T

Figure A.0.: Spectrogram analysis of audio signals from HolybroX500 and Y6Areiom UAVs at 20% speed (cont.)

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-21-01 Marco Stephan, Batbayar Battseren, Wolfram Hardt, UAV Flight using a Monocular Camera, März 2021, Chemnitz

CSR-21-02 Hasan Aljzaere, Owes Khan, Wolfram Hardt, Adaptive User Interface for Automotive Demonstrator, Juli 2021, Chemnitz

CSR-21-03 Chibundu Ogbonnia, René Bergelt, Wolfram Hardt, Embedded System Optimization of Radar Post-processing in an ARM CPU Core, Dezember 2021, Chemnitz

CSR-21-04 Julius Lochbaum, René Bergelt, Wolfram Hardt, Entwicklung und Bewertung von Algorithmen zur Umfeldmodellierung mithilfe von Radarsensoren im Automotive Umfeld, Dezember 2021, Chemnitz

CSR-22-01 Henrik Zant, Reda Harradi, Wolfram Hardt, Expert System-based Embedded Software Module and Ruleset for Adaptive Flight Missions, September 2022, Chemnitz

CSR-23-01 Stephan Lede, René Schmidt, Wolfram Hardt, Analyse des Ressourcenverbrauchs von Deep Learning Methoden zur Einschlagslokalisierung auf eingebetteten Systemen, Januar 2023, Chemnitz

CSR-23-02 André Böhle, René Schmidt, Wolfram Hardt, Schnittstelle zur Datenakquise von Daten des Lernmanagementsystems unter Berücksichtigung bestehender Datenschutzrichtlinien, Januar 2023, Chemnitz

CSR-23-03 Falk Zaumseil, Sabrina, Bräuer, Thomas L. Milani, Guido Brunnett, Gender Dissimilarities in Body Gait Kinematics at Different Speeds, März 2023, Chemnitz

CSR-23-04 Tom Uhlmann, Sabrina Bräuer, Falk Zaumseil, Guido Brunnett, A Novel Inexpensive Camera-based Photoelectric Barrier System for Accurate Flying Sprint Time Measurement, März 2023, Chemnitz

CSR-23-05 Samer Salamah, Guido Brunnett, Sabrina Bräuer, Tom Uhlmann, Oliver Rehren, Katharina Jahn, Thomas L. Milani, Güunter Daniel Rey, NaturalWalk: An Anatomy-based Synthesizer for Human Walking Motions, März 2023, Chemnitz

CSR-24-01 Seyhmus Akaslan, Ariane Heller, Wolfram Hardt, Hardware-Supported Test Environment Analysis for CAN Message Communication, Juni 2024, Chemnitz

Chemnitzer Informatik-Berichte

CSR-24-02 S. M. Rizwanur Rahman, Wolfram Hardt, Image Classification for Drone Propeller Inspection using Deep Learning, August 2024, Chemnitz

CSR-24-03 Sebastian Pettke, Wolfram Hardt, Ariane Heller, Comparison of maximum weight clique algorithms, August 2024, Chemnitz

CSR-24-04 Md Shoriful Islam, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Predictive Learning Analytics System, August 2024, Chemnitz

CSR-24-05 Sopuluchukwu Divine Obi, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Frontend for Agents in a Virtual Tutoring System, August 2024, Chemnitz

CSR-24-06 Saddaf Afrin Khan, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Diagnostic Learning Analytics System, August 2024, Chemnitz

CSR-24-07 Túlio Gomes Pereira, Wolfram Hardt, Ariane Heller, Development of a Material Classification Model for Multispectral LiDAR Data, August 2024, Chemnitz

CSR-24-08 Sumanth Anugandula, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Virtual Agent for Interactive Learning Scenarios, September 2024, Chemnitz

CSR-25-01 Md. Ali Awlad, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, AUTOSAR Software Component for Atomic Straight Driving Patterns, März 2025, Chemnitz

CSR-25-02 Billava Vasantha Monisha, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, Automotive Software Component for QT Based Car Status Visualization, März 2025, Chemnitz

CSR-25-03 Zahra Khadivi, Batbayar Battseren, Wolfram Hardt, Acoustic-Based MAV Propeller Inspection, Mai 2025, Chemnitz

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz