Reliability test and improvement of a sensor system for object detection

Course Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth and Dr. Andreas Pech

Richa Padhi
Mat No: 1438260
richa.padhi@stud.fra-uas.de

Theertha Bharathan
Mat No:1445457
theertha.bharathan@stud.fra-uas.de

***Abstract*—This study introduces a novel approach to enhance echo detection reliability in ultrasonic sensing systems through the integration of machine learning algorithms. This report delves into an experimental study that leverages the capabilities of ultrasonic sensors, specifically the Red Pitaya STEM Lab board and the Ultrasonic Sensor SRF02, to discern the position of the first echo in ultrasonic wave interactions with various objects. The research employs a multifaceted signal analysis methodology, integrating Ultrasonic Sensors (FIUS) for surface characterization, alongside advanced signal processing techniques such as FFT and the Hilbert Transform. The core of the analysis involves a comparative evaluation of machine learning models—Convolutional Neural Networks (CNN), Random Forest, and XGBoost—trained on ultrasonic signal data to classify object types and measure distances accurately. Additionally, enhancements to the ultrasonic measurement system are outlined, focusing on decision speed, accuracy, and usability improvements through optimized algorithms, advanced signal processing, and GUI development. This study not only advances the understanding of ultrasonic signal interactions but also sets the stage for future innovations in sensor technology and machine learning applications in non-destructive testing, medical imaging, and autonomous navigation systems.**

Keywords—Red Pitaya, Ultrasonic Sensor SRF02, Fast Fourier Transform, Machine Learning, Supervised Learning, Convolutional Neural Networks, Random Forest, XGBoost.

I. INTRODUCTION

In today's evolving world of technology, there is a focus, on improving the reliability and accuracy of sensor systems in various fields ranging from automotive safety to industrial automation. One of the challenges faced in this area is the ability to differentiate between objects and people accurately using sensor data in the context of driver assistance systems in vehicles. With advancements towards features in the automotive sector the need for dependable sensor systems that can distinguish between stationary obstacles and pedestrians is becoming increasingly critical.

The foundation of sensor systems relies on analysis of ultrasonic signals, which offer crucial insights into the surrounding environment. By employing signal processing techniques tailored to requirements these systems aim to extract valuable information from reflected signals to facilitate informed decision-making processes essential for ensuring safe and efficient vehicle operations.

Our research focuses on enhancing sensor system capabilities by distinguishing between living objects and human beings. By utilizing a range of signal analysis methods including Fourier transforms and custom approaches we aim to maximize the potential of sensor data, for accurate object detection.

Our sensor system architecture encompasses a synergistic blend of hardware and software components, each meticulously designed to fulfil specific roles in the signal processing pipeline. From the ultrasonic transducer responsible for emitting and receiving acoustic waves to the embedded control system orchestrating signal acquisition, and finally, the computational unit executing advanced analysis algorithms, every element of the system contributes to the overarching goal of enhancing detection reliability.

Critical to the success of our research endeavours is the rigorous collection and analysis of empirical data sets, meticulously curated to encompass diverse environmental conditions and object scenarios. Through iterative refinement and validation processes, we endeavour to iteratively enhance the accuracy and dependability of our sensor system outputs, ensuring their suitability for real-world deployment scenarios.

Furthermore, our research extends beyond mere data analysis, delving into the realm of machine learning methodologies to unlock deeper insights and capabilities within the sensor system framework. By harnessing the power of supervised learning algorithms, we aim to construct predictive models capable of discerning subtle patterns within the sensor data, thereby further augmenting the system's ability to accurately differentiate between objects and humans.

In parallel with our empirical investigations, we embark on the development of software solutions engineered to automate and streamline the distance calculation process, leveraging proprietary algorithms for first echo detection.

This synthesis of theoretical research and practical application heralds a new era in sensor technology, where machine learning techniques converge with traditional ultrasonic distance measurement methodologies to deliver unparalleled levels of accuracy and reliability.

Through the culmination of these multifaceted research endeavours, our aim is to push the boundaries of what is achievable with ultrasonic sensor technology, paving the way for safer, more efficient vehicular operations, and unlocking new possibilities in a myriad of industrial applications. As we navigate the intricate interplay between theory and practice, hardware and software, our journey towards enhancing sensor system reliability and precision stands as a testament to the relentless pursuit of innovation in the realm of autonomous systems and intelligent technologies.

II. METHODOLOGY

The theoretical background of the experiment is mentioned in the above section which consists of the description of the Ultrasonic Red Pitaya sensor, FFT data analysis, Machine Learning algorithm background.

A. Ultrasonic sensor and Red Pitaya Measurement Board

A test and measurement board called Red Pitaya STEM Lab [1] is based on a system-on-a-chip (SoC) [2] from the former company Xilinx. It may be configured to function as an oscilloscope, spectrum analyser, LCR meter, or network analyser and can be remotely controlled. The Ultrasonic Sensor SRF02 [3], a single transducer ultrasonic rangefinder in a tiny footprint PCB, was utilized in this configuration. The minimum range of the SRF02 is greater than that of other dual transducer rangers since it only employs one transducer for transmission and receiving. The smallest measuring range is approximately 15 cm (6 inches). With a 5V grounded power supply, it can operate. The Red Pitaya device makes it possible to wirelessly transfer data to a laptop for additional processing.

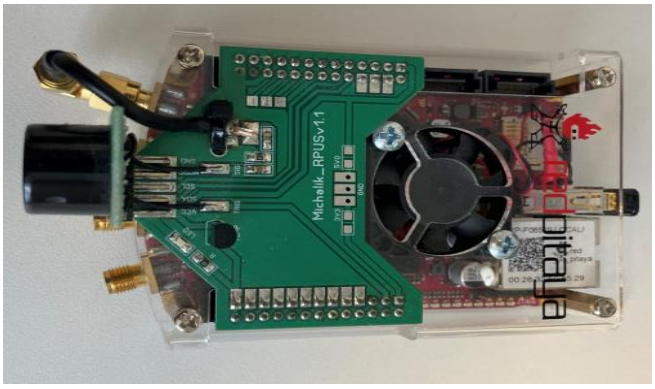

Fig.1. Ultrasonic sensor and red pitaya device

The sensor is operated under the GNU/Linux operating system. On a computer or mobile device, they can be used to manage and record measurements. In addition to 16 standard input and output ports, the main Red Pitaya unit incorporates two analog RF inputs and outputs. A micro-SD card slot, an RJ45 socket for Ethernet, a USB port, and a micro-USB connector for the console are also included on the board. Radio frequency transmissions can both be received and transmitted by the Red Pitaya which operates in the frequency range of 50MHz.

B. Theory of Ultrasonic Object Differentiation

Object differentiation with ultrasound involves using ultrasonic waves to distinguish between different objects or materials based on their physical properties, such as density, composition, and structure. This technique is commonly used in various fields, including medical imaging, non-destructive testing, and industrial inspection.

Ultrasound works on the principle of sending high-frequency sound waves into a material or object and analysing the echoes that bounce back. When an ultrasonic wave encounters a boundary between two different materials (e.g., air and tissue, metal, and plastic), part of the wave is reflected back while the rest continues to penetrate deeper into the material. By analysing the time, it takes for the echoes to return and their amplitude, it's possible to determine the properties of the materials and differentiate between them.

Physical Phenomena and Their Impacts:

1. Attenuation: Attenuation refers to the gradual loss of intensity of the ultrasound signal as it travels through a medium. This phenomenon occurs due to absorption, scattering, and reflection of the sound waves by the material. Attenuation can vary depending on factors such as the frequency of the ultrasound waves, the properties of the material, and the distance travelled. High levels of attenuation can reduce the reliability and precision of ultrasonic object differentiation, as it may result in weaker echoes and decreased signal-to-noise ratio [4].

2. Reflection and Refraction: When an ultrasound wave encounters a boundary between two materials with different acoustic impedances (the product of density and sound velocity), part of the wave is reflected back, and part is transmitted into the second material. The angle of reflection and refraction depends on the acoustic properties of the materials involved. Reflection and refraction can affect the accuracy of object differentiation, especially at interfaces between materials with significant differences in acoustic impedance [5].

3. Scattering: Scattering occurs when ultrasound waves interact with small irregularities or inhomogeneities within a material, causing the wavefront to deviate from its original path. Scattering can result in diffuse reflections and a loss of coherence in the received signal. In materials with high levels of scattering, such as biological tissues, object differentiation may be more challenging due to the complexity of the echo patterns.

4. Multipath Propagation: Multipath propagation refers to the phenomenon where ultrasound waves travel along multiple paths between the transmitter and receiver, leading to the reception of multiple echoes from the same object. This can complicate the interpretation of the received signal and make it more challenging to differentiate between objects, especially in environments with complex geometries or structures [6].

5. Noise: Noise sources, such as electronic interference, environmental factors, and system artifacts, can introduce additional signals into the received ultrasound data, leading to false detections or reduced signal quality. Minimizing noise and optimizing signal processing algorithms are essential for improving the reliability and precision of object differentiation with ultrasound.

In summary, while ultrasound offers valuable capabilities for object differentiation, various physical phenomena such as attenuation, reflection, scattering, multipath propagation, and noise can impact the reliability and precision of the technique. Understanding these phenomena and their effects is crucial for developing effective ultrasonic sensing systems and interpreting ultrasound data accurately in practical applications.

C. Theoretical Framework and Signal Analysis Methodology for Ultrasonic Sensor Systems

This research explores the effectiveness of Frequency-Modulated Integrated Ultrasonic Sensors (FIUS) in distinguishing between inanimate objects and humans by examining the ultrasonic backscatter signals. It's predicated on the notion that the unique surface features of different entities alter the reflected ultrasonic signals in distinct ways, enabling their identification based on the characteristics of these signals. The FIUS sensor works by emitting ultrasonic waves, typically around 40 kHz, and analysing the frequency spectrum of the signals that bounce back from the target. The variation in the reflected signals, influenced by the material and texture of the target's surface, results in discernible changes in the spectrum of the received signals. When these ultrasonic waves are emitted, their interaction with the target's surface generates a backscattered signal that encodes the surface attributes [7].

Surface Characterization Through Signal Analysis:

- *Signal Modulation by Surface Texture:* The study theorizes that smooth surfaces, such as car bumpers, reflect ultrasonic waves more uniformly, resulting in a backscattered signal with a smoother frequency spectrum. In contrast, more complex surfaces, like clothing on pedestrians, cause diffraction and scattering of the ultrasonic waves, leading to a frequency spectrum with multiple peaks and a less uniform distribution.

- *Frequency Shift Analysis:* The analysis focuses on detecting shifts in the peak frequencies of the backscattered signal. A shift from the mean frequency of the emitted signal indicates interaction with non-uniform surfaces, providing a basis for distinguishing between different types of objects.

- *Spectral Shape Examination:* Beyond frequency shifts, the overall shape of the signal's frequency spectrum provides additional insights into the surface's texture. The study posits that non-Gaussian spectral shapes are indicative of complex surfaces, such as those encountered in pedestrian clothing.

The research methodology involves a comparative analysis of the frequency spectra from the backscattered signals off known solid and soft targets. This comparison is intended to uncover specific spectral patterns and markers linked to various surface types. Statistical methods are employed to measure these differences and validate the spectral features' effectiveness as indicators of surface texture [7].

The anticipated results of this theoretical investigation are set to enhance the comprehension of how ultrasonic signals interact with different surfaces, aiding the development of more refined and precise pedestrian detection technologies. By exploiting the subtle variations in ultrasonic backscatter, this approach seeks to improve object recognition capabilities, potentially benefiting a wide range of applications [7].

D. Short Literature overview on ultrasonic distance measurement

Ultrasonic sensors find extensive applications in distance measurement owing to their dependability, precision, and adaptability. These sensors employ ultrasonic waves, characterized by frequencies beyond the upper threshold of human hearing, usually exceeding 20 kHz, to ascertain the distance between the sensor and an object. An overview of distance measurement methods using ultrasonic sensors:

1) Time-of-Flight (ToF) Principle:

Basic Concept: Ultrasonic sensors emit a burst of ultrasonic waves, and the time it takes for the waves to travel to the target object and back is measured. The Time-of-Flight (ToF) principle is a method commonly used in various applications, particularly in the field of distance measurement and imaging. This principle relies on measuring the time it takes for a signal or wave to travel from a source to a target and back again.

In the context of distance measurement, such as with ToF sensors or LiDAR (Light Detection and Ranging) systems, the ToF principle involves sending a signal, often a light pulse or an ultrasonic wave, toward a target. The sensor then measures the time it takes for the signal to travel to the target and be reflected to the sensor. [8]

Distance Calculation: The distance (D) is calculated using the formula: D = (Speed of Sound × Time) / 2.

Accuracy: ToF-based ultrasonic sensors can provide high accuracy in distance measurements.

2) Pulse-Echo Method:

Working Principle: Ultrasonic sensors generate short ultrasonic pulses and measure the time it takes for the pulse to travel to the object and return as an echo.

In ultrasonic testing, such as in medical imaging or non-destructive testing of materials, the Pulse-Echo method involves the following steps:

- *Pulse Transmission:* A short burst of ultrasonic waves is generated and directed toward the object or material being examined.
- *Reflection:* When these waves encounter a boundary or interface within the material (due to a change in acoustic impedance), a portion of the waves reflects towards the source.

- *Echo Reception:* A sensor or receiver detects the reflected waves, known as echoes, and measures the time it takes for them to return.
- *Distance Calculation:* The distance to the reflecting surface can be determined by multiplying the time of flight by the speed of sound in the material and dividing by two, as the signal travels to the object and back. [9]

Transducer: The sensor typically consists of a transducer that functions as both a transmitter and a receiver.

Application: Commonly used in industrial automation, robotics, and obstacle detection systems.

3) Phase Shift Measurement:

Basic Concept: The distance is measured by the phase shift between the transmitted and received ultrasonic waves. [10]

Advantages: This method can provide high resolution and is less affected by environmental conditions.

4) Multi-Echo Detection:

Working Principle: Utilizes multiple echoes from different surfaces to improve accuracy and reliability. [10]

The process of multi-echo detection typically includes the following steps:

- Pulse Transmission: A burst of ultrasonic waves is transmitted towards the target or object whose distance is to be measured.
- Echo Reception: The ultrasonic waves encounter surfaces or boundaries within the environment, leading to multiple echoes being reflected toward the sensor.
- Time-of-Flight Measurement: The time it takes for each echo to return to the sensor is measured. Using the time-of-flight principle, distances to the various reflecting surfaces can be calculated.
- Echo Analysis: Multiple echoes may be received due to reflections from different surfaces or objects. The sensor distinguishes between these echoes based on their respective time delays or time-of-flight measurements.
- Distance Calculation: The distances to each reflecting surface are calculated individually using the time-of-flight measurements. This information provides a comprehensive understanding of the spatial configuration of the environment.

Applications: Particularly useful in environments with multiple reflective surfaces or in applications where accurate measurements are critical.

5) Dual-Transducer Systems:

Setup: Involves separate transducers for transmitting and receiving ultrasonic waves. [9]

Benefits: Reduces the impact of signal crossover and enhances performance in challenging conditions.

6) Temperature Compensation:

Challenge: Ultrasonic wave speed is affected by temperature variations. [8]

Solution: Sensors may incorporate temperature sensors to compensate for temperature-induced changes in the speed of sound.

E. Hilbert Transform:

This method analyses the analytic signal obtained from the Hilbert transform of the data to detect peaks. It is effective for detecting peaks with varying widths and shapes. Hilbert Transform is a mathematical operation which provides a way to analyse the phase and amplitude of a complex signal. It is named after the German mathematician David Hilbert and used widely all over the world in signal domain now. The Hilbert transform can be a filter which simply shifts phases of all frequency components of its input by -π/2 radians [11]. The Hilbert transform of a function f(x) is defined by:

$$F(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(x)}{t-x}dx$$

Computationally it can be one can write the Hilbert transform as the convolution:

$$F(t) = \frac{1}{\pi t}\cdot f(t)$$

Which, by the convolution theorem of Fourier transforms, may be evaluated as the product of the transform of f(x) with -i*sgn(x), where:

$$\operatorname{sgn}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

The scientific computing software MATLAB has a Hilbert () function that "computes the so-called discrete-time analytic signal X = Xr + i*Xi such that Xi is the Hilbert transform of Xr" [11]. In MATLAB's implementation of the Hilbert () function takes advantage of the fast Fourier transform (FFT).

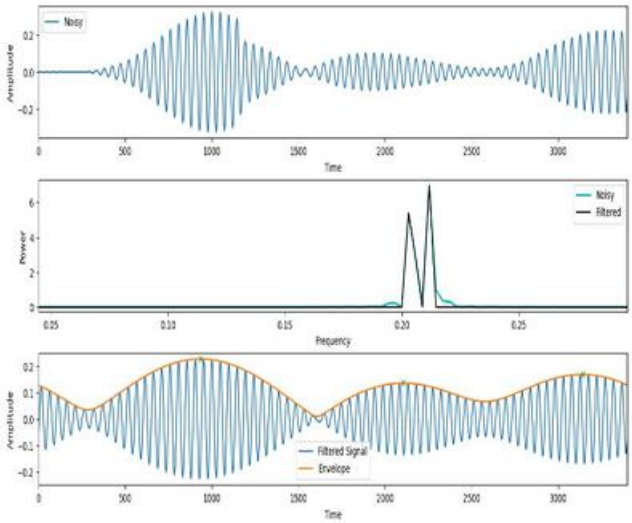

Fig.2. Upper enveloping by Hilbert Transform

The calculation is carried out in three phases by the Hilbert () function - Start by doing the FFT of Xr. Then, make all of the FFT elements that correspond to frequency $-\pi < \omega < 0$ to zero. The last step would be performing the Inverse FFT [12].

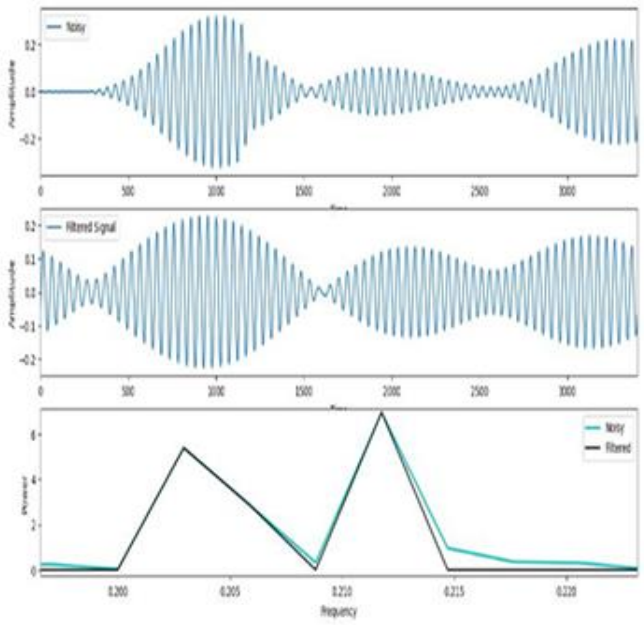

Fig.3. Filtering signal using FFT

Figure 3 illustrates how the Hilbert transformer is advantageous for extracting the envelope. This is explained by the formula that states the Hilbert transform yields a sin(t) for each cos(t) as HT provides a ±90° phase shift to the input signal. HT can produce the time signal's envelope by determining the time function's magnitude. The gradual overall change and sound intensity over time are analysed using envelopes.

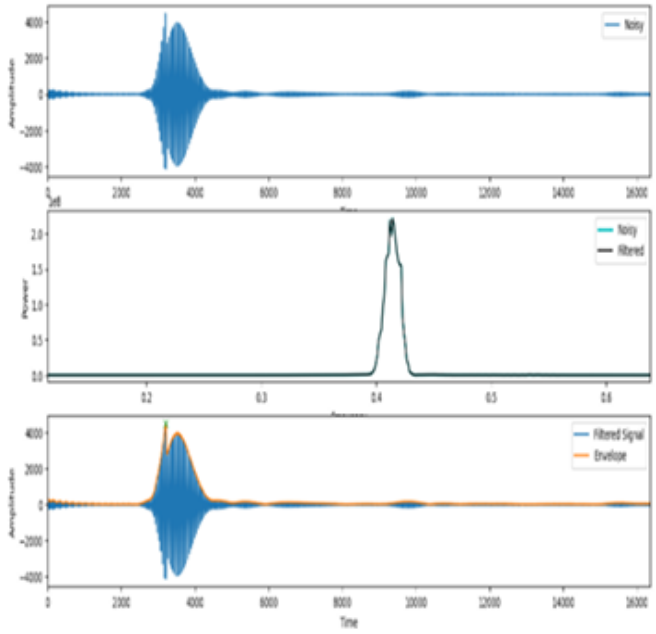

Fig.4. Performing FFT and Hilbert Transform

Since the real signal may be analytically extended from the real axis to the upper half of the complex plane, the Hilbert transform can be used to identify an imaginary function to a real valued signal. The outcome is as shown in figure 4.

- *ML method used for finding first reflection.*

Implementing a machine learning method for identifying the first reflection in an audio or acoustic environment involves several key steps. Below is a generalized approach that you can follow:

1. *Define the Problem*: Clearly define the problem and the objectives of identifying the first reflection. Understand the characteristics of first reflections in the context of your application. [13] [14]

2. *Data Collection*: Gather a dataset that includes audio recordings with labelled information about the presence and characteristics of first reflections. Ensure diversity in the dataset to capture various acoustic conditions. [13] [14]

3. *Data Preprocessing*:

- Audio Segmentation: Divide audio recordings into segments relevant to the analysis.
- Feature Extraction: Extract relevant features from the audio segments (e.g., spectral features, time-domain features, MFCCs).
- Labelling: Ensure accurate labelling of the dataset, marking segments with and without first reflections. [13] [14]

4. *Data Splitting*: Split the dataset into training, validation, and test sets to evaluate the model's performance [13] [14].

5. *Model Selection*: Choose an appropriate machine learning model based on the nature of the problem. Experiment with different algorithms such as SVM, decision trees, CNNs, or hybrid models depending on your dataset and problem requirements.

6. *Model Architecture*: Design the architecture of the chosen model, considering input features, hidden layers, and output layer. For deep learning models, experiment with architectures like CNNs or hybrid CNN-LSTM models [14].

7. *Feature Scaling and Normalization*: Standardize or normalize the input features to ensure that the model trains effectively and converges faster.

8. *Model Training*: Train the model using the training dataset. Fine-tune hyperparameters through iterative training and validation [13].

9. *Model Evaluation*: Evaluate the model's performance on the validation set to avoid overfitting. Adjust the model architecture or hyperparameters as needed [14].

10. *Testing*: Assess the final model on the test set to gauge its generalization performance.

11. *Post-Processing (if needed):* Implement any post-processing steps, such as filtering or thresholding, to refine the model's output.

12. *Interpretability and Visualization:* Analyse and visualize the model's predictions to gain insights into its decision-making process. Understand which features are most informative for detecting first reflections [14].

13. *Deployment*: If the model meets the desired performance, deploy it in the target environment. Consider real-time or batch processing depending on the application requirements [13].

14. *Continuous Monitoring and Improvement*: Implement a monitoring system to track the model's performance over time. Consider retraining the model periodically with new data to adapt to changing conditions. [13]

15. *Documentation*: Document the entire pipeline, including data preprocessing steps, model architecture, hyperparameters, and any insights gained during the development process.

As part of our project, we experimented with three models:

1. Convolutional Neural Networks (CNN)

1. *Data Preprocessing:* The dataset is pre-processed to ensure uniformity and suitability for input into the CNN model. This may involve tasks such as normalization, resizing, and augmentation to enhance the robustness and generalization capabilities of the model.

2. *Model Architecture:* A CNN architecture is designed, typically comprising multiple convolutional layers followed by pooling layers for feature extraction and dimensionality reduction, respectively. The final layers consist of fully connected layers for classification.

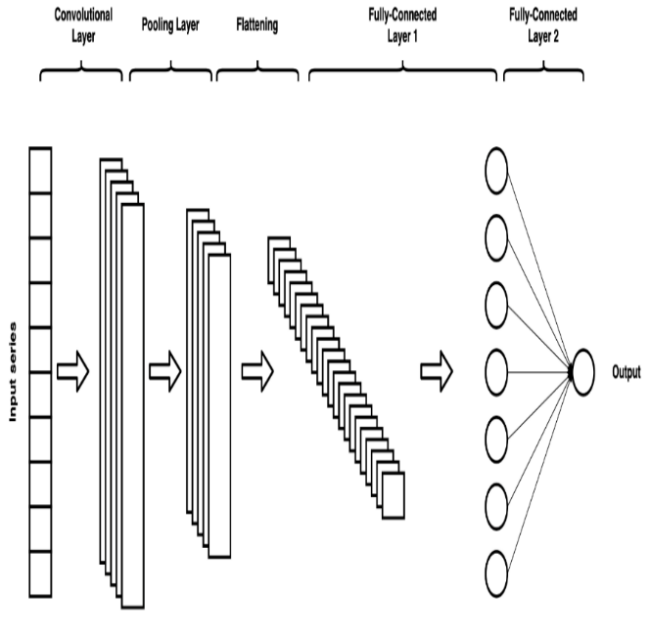

Fig.5. Simplified schema of 1D CNN model

3. Training: The CNN model is trained on the pre-processed dataset using backpropagation and optimization algorithms such as stochastic gradient descent (SGD) or Adam. During training, the model learns to extract hierarchical features from the input data, enabling it to discriminate between different classes [15].

4. Hyperparameter Tuning: Hyperparameters such as learning rate, batch size, and kernel sizes are tuned to optimize the performance of the CNN model and prevent overfitting.

5. Evaluation: The trained CNN model is evaluated on a separate test dataset to assess its performance metrics such as accuracy, precision, recall, and F1 score [16].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F1\text{-}Score = \frac{2\times Precision\times Recall}{Precesion+Recall}$$

2. Random Forest

1. Data Preparation: The dataset is prepared by splitting it into training and testing sets to facilitate model training and evaluation.

2. Ensemble Learning: A random forest ensemble model is constructed by combining multiple decision trees. Each tree is trained on a random subset of the training data and features, introducing diversity and reducing overfitting.

3. Training: The random forest model is trained on the training dataset using the bagging (bootstrap aggregating) technique, where each tree is trained independently.

4. Feature Importance: The importance of each feature in the dataset is assessed based on its contribution to the overall performance of the random forest model. This information can help identify relevant features and improve model interpretability [17].

5. Evaluation: The trained random forest model is evaluated on the test dataset to measure its performance in terms of accuracy, precision, recall, and other relevant metrics [18].

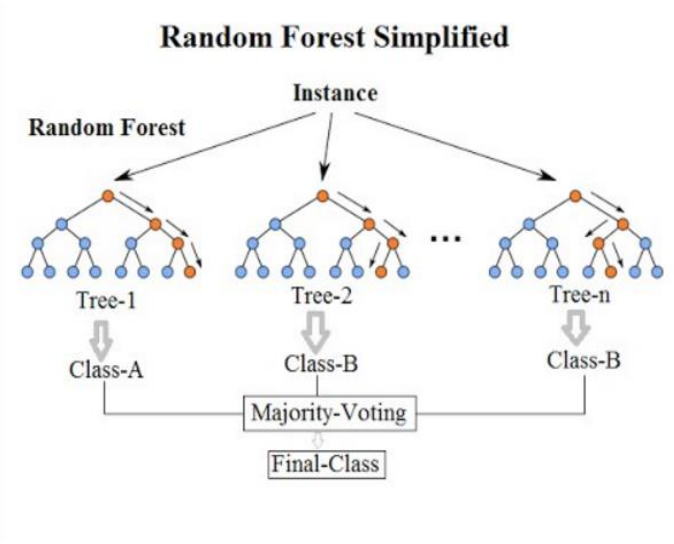

Fig.6.Random Forest

3. XGBoost

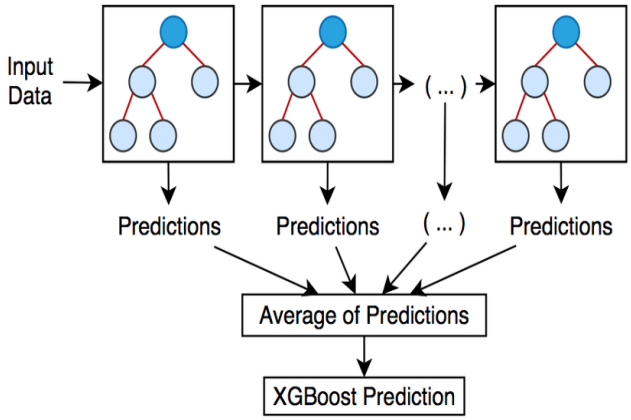

Fig.7. XGBoost

1. Data Preparation: Similar to other machine learning models, the dataset is prepared by splitting it into training and testing sets.

2. Model Initialization: An XGBoost model is initialized with default hyperparameters or manually specified hyperparameters based on domain knowledge and experimentation.

3. Training: The XGBoost model is trained on the training dataset using gradient boosting, where each subsequent tree is trained to correct the errors of the previous trees [19].

4. Hyperparameter Tuning: Hyperparameters such as learning rate, maximum depth of trees, and regularization parameters are tuned using techniques like grid search or random search to optimize model performance.

5. Evaluation: The trained XGBoost model is evaluated on the test dataset to assess its performance metrics such as accuracy, precision, recall, and F1 score [20].

III. IMPLEMENTATION

A. Measurement Environment and Setup

Laboratory Configuration: The experiments were carried out in the controlled setting of the Machine Learning laboratory at the Frankfurt University of Applied Sciences. The significance of a controlled environment is to minimize variables that could affect sensor accuracy.

Sensor and Object Placement: A FIUS sensor was strategically placed atop an elongated metal stand to facilitate a clear line of sight to the object of interest. Beneath it, the object—central to the experiment—was stationed on a white stand. This arrangement ensured consistent positioning for measurement repeatability.

Manual Verification Process: Distance measurements were manually recorded using a folding meter stick, serving a dual purpose: to calibrate the sensor and to provide a manual verification method for the sensor's readings. This process, vital for validating the accuracy of the sensor data, is documented in Figures 6 and 7.

Fig.8. Manually measuring distance using a folding meter stick

Fig.9. Manually measuring distance using a folding meter stick

B. Measurement Software Utilization

Software Interface and Functionality: The data from the Red Pitaya measurement board is collected using a Measurement software developed in Frankfurt University of Applied Sciences. The custom software interface, as detailed in Figure 8, was a critical tool in the data collection process. Its graphical user interface (GUI) facilitated real-time analysis and visualization of the sensor data. Users could choose between analysing raw analog data or applying Fast Fourier Transform (FFT) for frequency domain analysis.

Data Acquisition and Analysis: The software allowed for the export of FFT-transformed data in a text format, which could then be processed and analysed. Its capability to plot FFT graphs and time-series data provided a comprehensive understanding of the sensor's performance under various conditions.

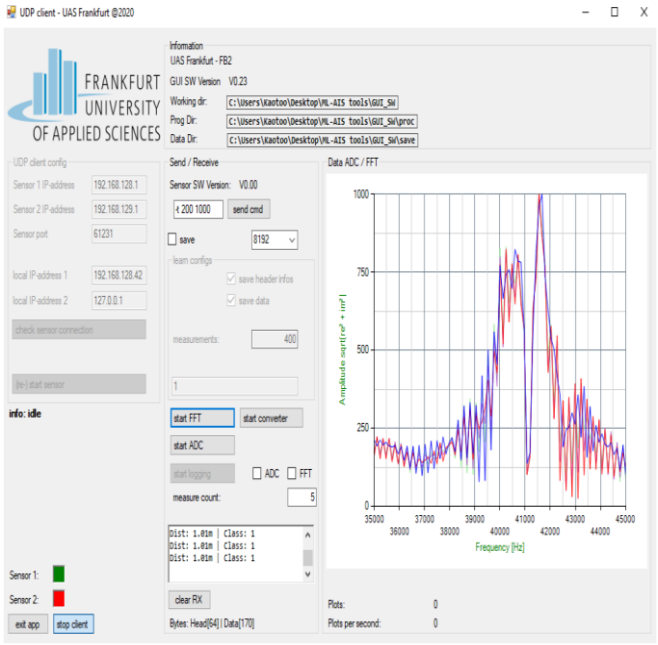

Fig.10. FFT data from the measurement software

C. *Data Structure and Export Format*

FFT Data and Headers:

For this measurement, we used text-based ADC data that was exported from the software. Each data has 16384 columns overall, with rows representing amplitude values in the range of 0 to 1000 and columns representing frequency values.

Data headers are also exported by the program in addition to ADC data. The length of the data, the classification outcome from the software's current model, or the sampling frequency are just a few examples of the useful information included in data headers. Figure 9 shows a sample data format, and the table below, Table I, provides information on the data headers.

64	32768	1	1.1	512	0	1953125	12	0	0.1	0.98	0.2	0.3 V0.2	0.4	0.5	-3	-4	-5	-1	-3.1
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-3	-2	1	-187	-155
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-9	-5	-3	-5	-232
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-11	-7	-6	-228	-248
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-2	-4	-6	-4	-216
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-5	-4	-6	-7	288
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-89	-76	-71	-71	-166
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-9	-8	-9	-14	-7
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	0	-1	-1	-9	-220
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	6	3	1	185	195
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-4	-1	-249	-228	-200
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-12	-14	-12	-262	-256
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	0	1	3	-113	-80
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	0	3	6	-161	-130
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-2	-4	-1	-22	23
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	2	-2	-2	-1	-21
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-4	-1	1	170	186
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-1	-2	1	173	184
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	5	-1	-2	-5	101
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-2	-2	-3	64	99
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-7	-8	-9	-9	183
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-5	-7	-8	-9	167
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-94	-91	-89	-167	-192
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	1	2	1	0	1
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-3	-6	-2	-224	-238
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	11	8	5	8	9
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	0	2	2	2	131
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0 V0.2	0	0	-1	-4	-2	36	74
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-10	-10	-12	-10	-255
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0 V0.2	0	0	-3	-7	-4	-7	-6

Fig.11. Raw measurement data

TABLE.I DESCRIPTION OF THE HEADERS IN THE MEASUREMENT FILE

Column	Header Description	Value	Remarks
1	Header lengths	64	
2	Data length	170	
3	Class detected	1 or 2	1: Object 2: Human
4	Measurement type	0 or 1	0 : FFT 1 : ADC
5	Frequency resolution f or sampling time t depend on measurement type	119	
6	-	0	irrelevant
7	Sampling frequency (Hz)	1953125	
8	ADC resolution	12	
9		0.0	irrelevant
10	Distance between sensor and first object (round-trip-time in µs)	-	
11	FFT Window length	0	
12		0	irrelevant
13	software version (RP)	V0.2	

Data Collection:

Following the precise arrangement of the measurement environment, we extracted FFT measurement data as text files from the Red Pitaya board, which was equipped with an echo-based ultrasonic sensor. This stage involved meticulous calibration to establish baseline conditions for a comprehensive comparative analysis. Our objective was to capture a dataset that reflects a variety of real-world scenarios by varying distances, environmental conditions, and object materials. Hard objects placed at a distance of 1m and 50cm. 1,000 Analog-to-Digital Converter (ADC) data points were systematically collected for each type of object. This process was repeated across two experimental runs to ensure data robustness and reliability. The extensive collection of data points was crucial to train our model effectively, allowing it to recognize and adapt to different scenarios.

- Detection of Hard-Object:

A rigid box was the chosen object for this phase due to its reflection characteristics, which are significantly different from organic materials. We positioned this box at two distances—1 meter and 50 centimetres—from the sensor. Each placement was carefully measured with a standard measuring scale to ensure accuracy. Any discrepancies between manual measurements and sensor readings were recorded and analysed using the UDP client application. This process helped in refining our model for distance estimation based on echo delay.

- Detection of Soft-Object/Pedestrian:

For the soft object detection, we simulated pedestrian scenarios by instructing an individual to stand and sit beneath the sensor at measured distances. This phase was designed to test the sensor's capability to detect and differentiate between inanimate objects and humans, an essential feature for applications such as automated surveillance or navigation systems.

Fig.12. Detection of hard-object and Fig.13. Detection of soft-object

D. *Data Preprocessing*

The implementation outlines a structured approach to processing and analysing ultrasonic signal data for the purpose of echo detection. Each step, from data preprocessing and signal windowing to noise reduction, peak detection, and data labelling, plays a crucial role in enhancing the signal's quality and extracting meaningful features. This thorough process ensures that the machine learning model is trained on high-quality, relevant data, thereby maximizing its effectiveness and accuracy in detecting the position of the first echo in ultrasonic signals.

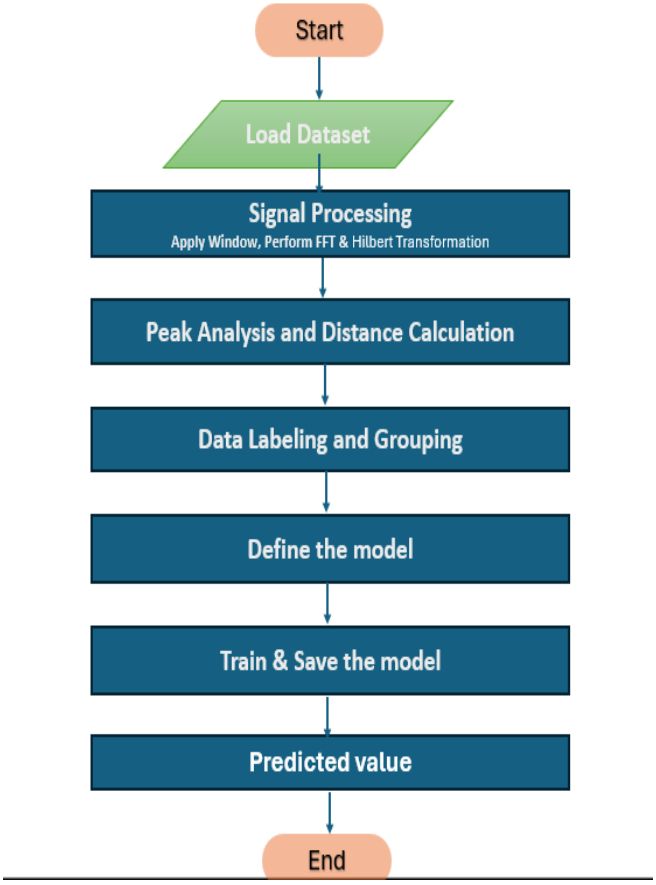

Fig.14. Flowchart of the Model

The implementation of the above methodology for creating a model to detect the position of the first echo is illustrated with the help of above Figure 14, describing the Flow chart for the Data Pre-Processing, Creation, Training and Prediction of the model for the project.

Software Implementation:

The implementation involves a comprehensive process for developing a machine learning model aimed at detecting the position of the first echo in ultrasonic signal data. This model is part of a larger system designed to interpret and analyse ultrasonic signals for various applications, such as non-destructive testing or medical imaging. The methodology encompasses several key stages, including data preprocessing, feature extraction, model creation, training, and prediction. This section focuses on elucidating each step in detail, providing a clear understanding of the underlying processes and the rationale behind them.

Data Pre-Processing and Feature Extraction Steps:

Data Preprocessing: The initial phase involves reading and preparing the data for subsequent analysis. This step is crucial as it sets the foundation for the model's accuracy and reliability. The process begins with importing signal data from a CSV file. Given that not all columns in the file may be relevant to the analysis, a selection is made to focus on specific columns containing the essential signal data. This refined dataset is then subjected to further preprocessing to ensure it is in a suitable format for analysis.

Signal Windowing: Signal windowing is a pivotal preprocessing step designed to mitigate spectral leakage, a common issue in signal processing where energy from the signal 'leaks' into adjacent frequencies. To address this, a windowing function, specifically a Hanning window, is applied to each signal. The Hanning window, characterized by its smooth, sinusoidal shape, effectively tapers the signal at its beginning and end, thereby reducing spectral leakage. This process not only enhances the quality of the frequency analysis but also improves the model's ability to accurately identify echoes within the signal.

Noise Reduction and Peak Detection: The next step involves transforming each signal from the time domain to the frequency domain using the Fast Fourier Transform (FFT). This transformation allows for the analysis of the signal's frequency components, facilitating the identification and removal of noise through a Power Spectral Density (PSD) thresholding method. Noise components with a PSD below the threshold are filtered out, and the signal is then reconstructed in the time domain using an inverse FFT.

Following noise reduction, the signal's envelope is extracted using the Hilbert Transform. The envelope, representing the signal's amplitude modulation, aids in the detection of peaks within the signal. Peak detection is a critical component of this process, as the position of peaks corresponds to potential echoes in the ultrasonic signal. The detection of these peaks

enables the identification of significant features within the signal, which are indicative of the presence and location of objects or structures being analysed by the ultrasonic system.

E. Labelling the data

This function is designed to process a list of detected peak positions from signal data and organize these findings into a structured format that indicates the presence of peaks within specified windows of time across multiple signals. It operates by first determining the number of possible windows in each signal, which is calculated by dividing the total length of the signal by the width of each window. The function then creates a matrix, to hold binary labels for each signal across all the windows.

For each signal, the function examines the peak position. If a peak is detected (indicated by a peak position greater than or equal to zero), the function calculates which time window this peak falls into. This is done by dividing the peak position by the window width.

Once the appropriate window for a peak is identified, the function sets the corresponding element in the label matrix to 1. This indicates that within this particular window of the signal, a peak is present. If a window contains no peaks, it remains labelled as 0, indicating the absence of a peak.

Ultimately, the function returns this label matrix, providing a simple, binary, structured representation of where peaks occur within the set of signals, as delineated by the predefined time windows. This labelled data matrix serves as a simplified, yet effective, representation of the signal's characteristics, enabling the machine learning model to learn from and recognize patterns associated with the detected peaks.

F. Training the Model

In the realm of machine learning, the choice of model architecture plays a pivotal role in the success of any data-driven project. This section delves into three advanced machine learning models - Convolutional Neural Networks (CNNs), Random Forest, and XGBoost - each known for their unique strengths in handling complex datasets with high-dimensional features. From capturing spatial and temporal dependencies in data to robust ensemble learning techniques, these models represent the cutting edge in predictive analytics and classification tasks. The following subsections provide an in-depth exploration of the initialization, training, evaluation, and practical implementation of these models, offering insights into their operational dynamics and effectiveness in real-world applications.

1. Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a class of deep neural networks, highly effective for analysing data with spatial hierarchies or temporal sequences. Their ability to capture spatial and temporal dependencies in data through the application of filters makes them particularly suitable for image and time-series data analysis.

a) Model Initialization and Configuration

The CNN model's training begins with setting key parameters that govern the training process. The verbose parameter controls the level of detail logged during training, with 1 indicating progress output for each epoch. The epochs parameter, set to 10, defines the total number of complete passes through the entire training dataset. The batch size of 32 specifies the number of samples per gradient update, balancing training speed and model update granularity.

b) Model Architecture

The model architecture is constructed using a sequential approach, allowing layers to be added in a linear fashion. The core of the CNN model includes:

- Conv1D Layers: Two convolutional layers with 64 filters and a kernel size of 3 are employed to perform convolution operations on the 1D input data. These layers extract high-level features by applying filters across the time series data. The relu activation function introduces non-linearity, enabling the model to learn complex patterns.
- Dropout Layer: A dropout layer with a rate of 0.5 is introduced to prevent overfitting by randomly setting a fraction of input units to 0 during training. This encourages the model to learn more robust features that are not reliant on any small set of neurons.
- MaxPooling1D Layer: This layer reduces the dimensionality of the data by pooling over temporal dimensions, which helps in reducing the number of parameters and computational cost, and controls overfitting.
- Flatten Layer: The Flatten layer converts the 2D feature maps produced by the convolutional and pooling layers into a 1D vector. This is necessary to transition from convolutional layers to dense layers, which require 1D input.
- Dense Layers: Following the Flatten layer, a dense layer with 1052 units employs the relu activation function to introduce further non-linearity and learning capacity. The final dense layer uses a softmax activation function to output a probability distribution over the target classes, facilitating multi-class classification.

c) Model Compilation

The model is compiled with the categorical_crossentropy loss function, suitable for multi-class classification tasks, and the adam optimizer, known for its efficiency and adaptive learning rate properties. The compilation step prepares the model for training by setting up the backend configuration, such as the loss function, optimizer, and metrics to monitor.

d) Model Training

Model training is executed on the provided training data (xtrain, ytrain) over the defined number of epochs and batch size. This process involves forward propagation, loss computation, backpropagation, and weight updates iteratively to minimize the loss function. In cases where the dataset is imbalanced, assigning class weights can help the

model pay more attention to underrepresented classes during training. While not explicitly mentioned in the code, class weights can be calculated based on the class distribution and passed to the fit method via the class weight parameter to adjust the loss function during training.

e) Model Evaluation
Upon completion of training, the model's performance is evaluated on a separate test dataset (xtest, ytest) to assess its accuracy and generalization capability. The accuracy metric, expressed as a percentage, provides a quantitative measure of the model's ability to correctly predict the class labels of unseen data.

f) Model Saving
The trained model is saved to the file system in the Hierarchical Data Format version 5 (.h5) file. This format is a popular choice for storing large numerical datasets like the weights of a neural network, allowing the model to be reloaded and used without retraining.

2. Random Forest
Random Forest is an ensemble machine learning method renowned for its high accuracy, flexibility, and ease of use. It constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests are robust against overfitting, especially in cases where the dataset is large.

a) Hyperparameter Grid Setup
Before training the model, a comprehensive grid of potential hyperparameters is established to guide the search for the optimal model configuration. These hyperparameters include:

- n_estimators: The number of trees in the forest. Values considered are 50, 100, and 150, to balance between computational efficiency and model accuracy.
- max_features: The maximum number of features considered for splitting at each leaf node, with options 'sqrt' and 'log2' to test different feature subsets.
- max_depth: The maximum depth of each tree, including None (trees grow until all leaves are pure or contain less than min_samples_split samples), 10, 20, and 30, allowing for flexibility in tree complexity.
- min_samples_split: The minimum number of samples required to split an internal node, with options 2, 5, and 10, to prevent overfitting.
- min_samples_leaf: The minimum number of samples required to be at a leaf node, with values 1, 2, and 4, influencing the decision-making at the leaf level.
- bootstrap: Indicates whether bootstrap samples are used for building trees, with options True (bootstrap sampling) and False (the entire dataset is used to build each tree), affecting sampling variability.

b) Randomized Search for Hyperparameter Tuning
Utilizing RandomizedSearchCV, the model explores the defined parameter space randomly, rather than exhaustively, selecting the best combination based on cross-validated performance. This approach is efficient and effective, especially when dealing with many hyperparameters and limited computational resources. The search is configured to run for 100 iterations with a 3-fold cross-validation strategy, ensuring a robust search process.

c) Model Training with Optimal Parameters
Upon identifying the best hyperparameters, a new Random Forest model is instantiated with these settings and trained on the full training dataset. This step ensures that the model is as optimized as possible for the given data.

d) Model Evaluation
The trained model is then evaluated on both the training and test datasets to assess its performance. The F1 score, a harmonic mean of precision and recall, is calculated for both sets to provide a balanced measure of the model's accuracy and its ability to handle imbalanced classes.

e) Model Saving
The best-performing model is serialized and saved as a pickle file to the file system. This allows the model to be persisted for future use without the need for retraining.

3. XGBoost
XGBoost is a powerful gradient boosting machine learning library that is often used for structured or tabular data.

a) Model Initialization

- XGBoost Classifier: The function initializes an XGBoost classifier with a set of specified hyperparameters. These parameters are crucial as they directly influence the model's learning process and overall performance.
- objective="binary:logistic": This specifies that the model is being trained for a binary classification task. The logistic function is used to predict the probability that a given input belongs to the class labeled "1".
- n_estimators=100: Defines the number of gradient-boosted trees to be constructed. More trees can lead to better performance but increase computational complexity.
- max_depth=3: Sets the maximum depth of each tree. Deeper trees can model more complex patterns but may lead to overfitting.
- learning_rate=0.01: Determines the step size at each iteration while moving toward a minimum of the loss function. A smaller learning rate requires more iterations but can lead to a more precise model.
- subsample=0.5: Indicates the fraction of samples to be randomly sampled for each tree. Subsampling prevents overfitting by making the model more robust.
- verbosity=0: Controls the level of verbosity of the output generated by the model during training. A value of 0 suppresses most of the output, making the training process cleaner.

b) Model Training

The fit method is used to train the model on the training dataset (xtrain, ytrain). During this phase, the model learns to map the input features to the target variable by minimizing the loss function over the series of trees.

c) Model Evaluation

The trained model is then used to make predictions on both the training (xtrain) and test datasets (xtest). The F1 score, a harmonic mean of precision and recall, is calculated for both sets of predictions. The average="weighted" parameter is used to calculate metrics for each label, and find their average weighted by the number of true instances for each label. This is useful for dealing with imbalanced datasets. The F1 scores are printed for both the training and test datasets, providing insight into the model's performance and its ability to generalize to unseen data.

d) Model Saving

After training, the model is serialized and saved to a file using the pickle protocol. This step is crucial for model deployment, as it allows the trained model to be stored and later loaded without retraining.

G. Implementation of GUI Interface

The implemented system encompasses both a signal processing component and a web application for interactive signal analysis. The signal processing component utilizes Python libraries such as NumPy, pandas, Matplotlib, and SciPy to process and visualize signals stored in data files. It performs tasks including signal transformation, filtering, and peak detection, generating plots for visualization of processed signals. Meanwhile, the web application is built using Flask, along with Flask-WTF for form validation and TensorFlow for model integration. The application provides a user-friendly interface for uploading data files and displaying analysis results. It utilizes a pre-trained model for signal analysis and can call external scripts for additional processing. The HTML template renders the interface, allowing users to upload files and view analysis outcomes dynamically. Configuration options are available for customizing paths and parameters, enhancing the system's adaptability across different environments. In summary, the system offers a comprehensive solution for signal analysis, combining standalone processing capabilities with interactive web-based visualization for intuitive user interaction and interpretation of results.

IV. RESULT AND ANALYSIS

A. Signal Processing Characteristics for Hard and Soft Objects

Our experimental outcomes have uncovered significant signal processing distinctions between hard and soft objects. With hard surfaces, the signals demonstrate smooth, Gaussian-like envelopes indicative of singular, robust reflections typical of rigid, reflective materials. Specifically, the data for hard surfaces placed at distances of 50 centimetres and 1 meter display pronounced peaks in the time domain. Their Fourier spectra feature dominant frequency components, suggesting minimal dispersion of the ultrasonic waves.

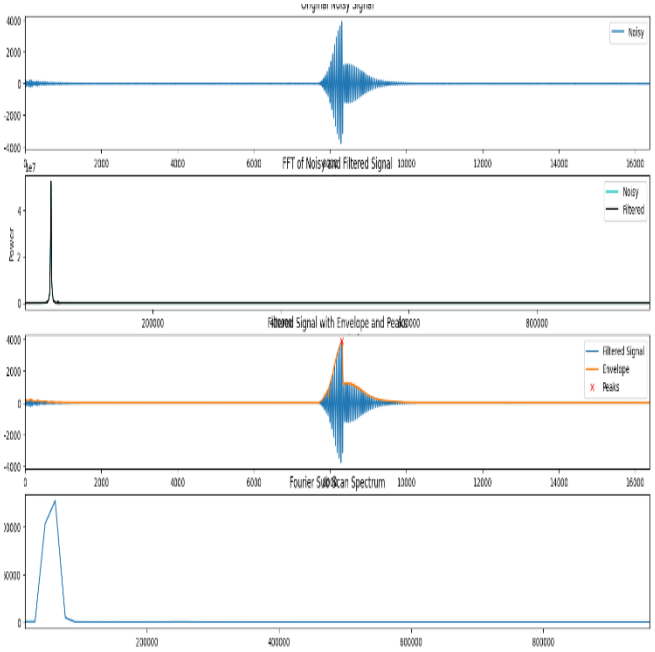

Fig.15. ADC to FFT plot for object placed at 1m.

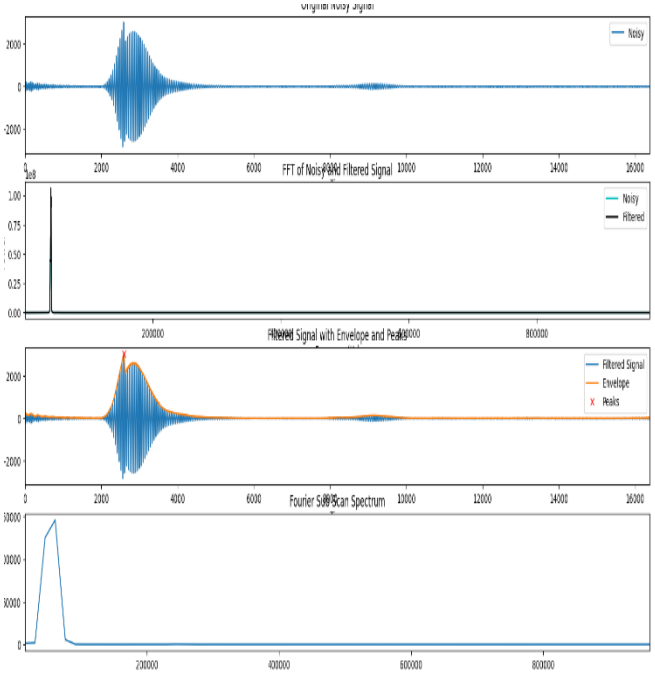

Fig.16. ADC to FFT plot for object placed at 50cm.

In contrast, soft objects, such as a person standing or sitting, generate signals with more complex spectral profiles, characterized by multiple peaks and a less defined envelope. This arises due to the ultrasonic waves scattering upon interaction with the varied shapes and absorptive qualities inherent in soft materials, leading to a broader spectrum with multiple frequency components.

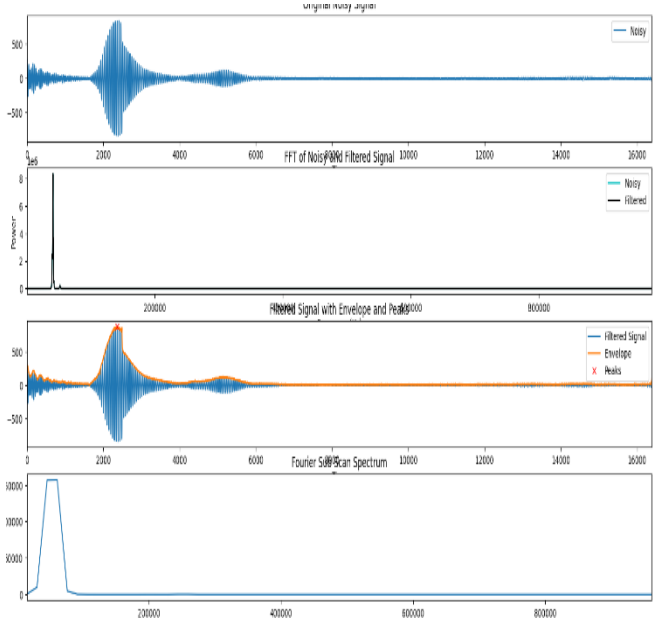

Fig.17. ADC to FFT plot for soft object standing.

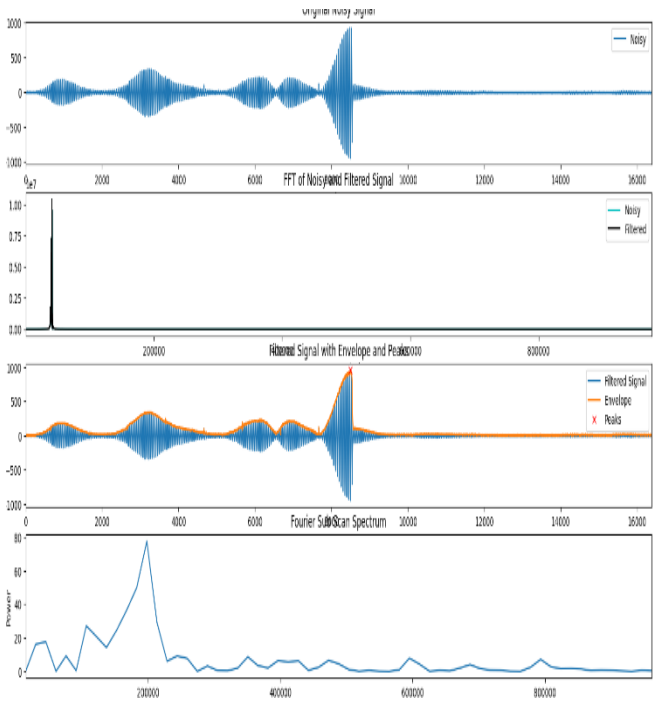

Fig.18. ADC to FFT plot for soft object sitting.

Fig.17. ADC to FFT plot for a soft object (standing person) and Fig.18. ADC to FFT plot for a soft object (sitting person) demonstrate the irregular envelopes and spectral plots with multiple peaks, indicative of the diverse echo signatures from soft targets.

These distinctive reflection signatures are critical for enhancing the sensor system's accuracy in discerning between object types, a capability crucial for applications in autonomous navigation and safety systems.

B. Sensor System's Efficacy in Distance Measurement

The precision of our sensor system in measuring distances was critically evaluated by determining the distance to a hard object placed one meter away from the sensor, using a sampling frequency of 1,953,125 Hz. The system consistently approximated the actual distance of 1 meter, with a mean calculated distance of approximately 1.446 units and a negligible standard deviation, indicating high precision.

The peak positions were identified within a narrow range of sample points, demonstrating the algorithm's reliable and consistent performance, and underscoring the system's robustness for precise distance measurements.

```
Row 0: Distance = 1.4454952960000003 units, Peak Position = 8231
Row 1: Distance = 1.4588421120000001 units, Peak Position = 8307
Row 2: Distance = 1.446724608 units, Peak Position = 8238
Row 3: Distance = 1.459544576 units, Peak Position = 8311
Row 4: Distance = 1.446548992 units, Peak Position = 8237
Row 5: Distance = 1.4632325120000003 units, Peak Position = 8332
Row 6: Distance = 1.446548992 units, Peak Position = 8237
Row 7: Distance = 1.4614763520000003 units, Peak Position = 8322
Row 8: Distance = 1.44268544 units, Peak Position = 8215
Row 9: Distance = 1.456559104 units, Peak Position = 8294
Row 10: Distance = 1.4463733760000002 units, Peak Position = 8236
```

Fig.19. Distance Calculation

C. Model Performance for Object Classification

Based on training the above 3 models for ADC data for hard objects at 1m. We could see that following results:

1. Convolutional Neural Networks (CNN)

The CNN model shows high precision, recall, and F1-score for the majority classes (128, 129, 130), indicating that it is predicting these classes very well. The overall accuracy is 0.905 (90.5% correct predictions). The weighted average of F1-score and recall are both also high at around 0.91, suggesting good performance across all classes while considering the number of instances (support) in each class.

```
Classification Report:
              precision    recall  f1-score   support

          92       0.00      0.00      0.00         1
         108       0.00      0.00      0.00         1
         122       0.00      0.00      0.00         1
         125       0.00      0.00      0.00         0
         126       0.00      0.00      0.00         4
         127       0.00      0.00      0.00         2
         128       0.95      0.99      0.97       107
         129       0.95      0.95      0.95        64
         130       0.82      0.90      0.86        20

    accuracy                           0.93       200
   macro avg       0.30      0.32      0.31       200
weighted avg       0.89      0.93      0.91       200

Accuracy: 0.925
Weighted F1 Score: 0.908613829093281
Weighted Recall: 0.925
```

Fig.20. Classification report of CNN Model

2. Random Forest

The Random Forest model, after hyperparameter tuning, achieved the best performance with 100 trees (n_estimators), a minimum of 5 samples required to split an internal node (min_samples_split), only 1 sample required at a leaf node (min_samples_leaf), max_features set to use the square root of the number of features, a maximum depth of the trees (max_depth) at 30, and without bootstrapping (bootstrap set to False).This model also performs well on the major classes, with slightly lower performance on class 130 compared to the CNN model. The overall accuracy of this model is higher at 0.925 (92.5% correct predictions). The weighted F1-score and recall are also high and consistent with the overall accuracy at around 0.92.

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
92	0.00	0.00	0.00	1
108	0.00	0.00	0.00	1
122	0.00	0.00	0.00	1
126	0.00	0.00	0.00	4
127	0.00	0.00	0.00	2
128	0.98	0.97	0.98	107
129	0.95	0.92	0.94	64
130	0.90	0.90	0.90	20
accuracy			0.91	200
macro avg	0.31	0.31	0.31	200
weighted avg	0.92	0.91	0.91	200

Accuracy: 0.905
Weighted F1 Score: 0.9121238542365303
Weighted Recall: 0.905

Fig.21. Classification report of Random Forest Model

3. XGBoost

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
92	0.00	0.00	0.00	1
108	0.00	0.00	0.00	1
122	0.00	0.00	0.00	1
126	0.00	0.00	0.00	4
127	0.00	0.00	0.00	2
128	0.95	0.98	0.97	107
129	0.96	0.80	0.87	64
130	0.94	0.85	0.89	20
accuracy			0.86	200
macro avg	0.32	0.29	0.30	200
weighted avg	0.91	0.86	0.89	200

Accuracy: 0.865
Weighted F1 Score: 0.8861899786687562
Weighted Recall: 0.865

Fig.22. Classification report of XGBoost Model

XGBoost model has precision, recall, and F1-scores similar to the Random Forest model for the major classes, with a very slight decrease in recall for class 130. The accuracy of the XGBoost model is 0.865 (86.5% correct predictions), which is lower than the CNN and Random Forest models. The weighted average F1-score and recall are correspondingly lower at around 0.88.

4. Merging Hard and Soft Objects Data for Enhanced CNN Model Training

Based on the evaluation of model performance for object classification, the Convolutional Neural Network (CNN) model emerged as the most effective for classifying hard objects at a distance of 1 meter, with significant accuracy. Encouraged by these results, we extended the application of the CNN model to a combined dataset that includes both hard and soft objects.

To prepare for this unified training approach, we first merged the ADC data from hard objects placed at 1m with that of soft objects in both standing and sitting positions. Recognizing the potential variances in scale and distribution across the two datasets, we employed a normalization step to standardize the data, ensuring a consistent range and scale. This normalization was achieved using the StandardScaler, which was applied to each feature independently, aligning the data distribution to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1. The training was conducted over 20 epochs with a batch size of 32, incorporating a learning rate scheduler to adapt the learning rate over time for optimal convergence.

The CNN model's adeptness at classifying ADC data for hard objects positioned at 1 meter was the catalyst for its application to a merged dataset. The unified dataset comprised 1000 instances each of hard objects, placed at 1 meter and 50 centimetres, and soft objects, in varying postures of sitting and standing. This approach was underpinned by the hypothesis that the CNN's demonstrated ability to discern distinctive signal processing characteristics would translate to robust performance on a more diverse set of data.

The classification report of this merged dataset is depicted in the image provided. It shows an overall accuracy of 63.61%, which reflects the average success rate of the model across all the different classes of objects in the dataset. Notably, this accuracy metric is a testament to the model's generalization ability when confronted with a larger and more complex dataset that blends various object types and distances.

Examining the specific details in the classification report, we can observe that certain classes, particularly those corresponding to hard objects, have maintained high precision and recall values similar to the model's previous exclusive analysis of hard objects. The model's capability to discern the nuanced differences between hard and soft objects

is evident, despite the overall reduction in accuracy when compared to the individual analyses of hard and soft objects.

This nuanced performance is attributed to the inherent challenge in classifying soft objects, which present more complex spectral signatures than hard objects. The varied shapes and materials of soft objects result in a broader spectrum of reflected ultrasonic waves, making classification a more intricate task. Consequently, while the model excels at identifying the more consistent signal patterns from hard objects, it encounters a greater challenge with the variable signatures of soft objects, as reflected by the diverse precision, recall, and F1-score metrics across classes.

Accuracy: 63.61%
23/23 ━━━━━━━━━━━━━━━━━━━━ 5s 200ms/step

	precision	recall	f1-score	support
35	0.67	0.50	0.57	8
36	0.65	0.69	0.67	32
37	0.51	0.68	0.58	34
38	0.40	0.33	0.36	24
39	0.00	0.00	0.00	8
40	0.51	0.59	0.54	75
41	0.00	0.00	0.00	6
45	0.71	0.62	0.66	104
46	0.70	0.76	0.73	25
113	0.00	0.00	0.00	5
127	1.00	0.25	0.40	4
128	0.98	0.99	0.98	126
129	0.95	0.92	0.94	65
130	0.82	0.64	0.72	22
131	0.00	0.00	0.00	9
132	0.00	0.00	0.00	9
133	0.33	0.05	0.09	19
134	0.27	0.52	0.35	25
135	0.51	0.58	0.54	45
136	0.55	0.70	0.61	50
137	0.21	0.33	0.26	12
138	0.12	0.10	0.11	10
139	0.00	0.00	0.00	10
140	0.00	0.00	0.00	3
152	0.00	0.00	0.00	1
accuracy			0.64	731
macro avg	0.40	0.37	0.37	731
weighted avg	0.62	0.64	0.62	731

Fig.23. Classification report of CNN Model combining both hard and objects

The 63.61% accuracy achieved in this expansive classification task, although modest, is indicative of a model that can still reliably distinguish between a wide range of object types in a complex dataset. The balance between precision and recall across classes, as represented by the F1-scores, emphasizes the model's applicative potential, especially in scenarios where differentiation between hard and soft objects is critical.

D. Impact of Window Size on Model Accuracy

A window size of 64 was utilized in our models, which directly impacts the accuracy of our classification system. While the precise window size deemed optimal may vary depending on the sampling rate and the specificity of the signals, it is important to align the window size with the kernel size of the CNN to optimize the feature extraction process. In our case, the selected window size does not perfectly match the actual peak but rather provides a window in which the peak is present.

The choice of a 64-unit window was a balance between capturing enough of the signal for accurate classification and maintaining computational efficiency. It is neither explicitly good nor bad without the context of the sampling rate; however, for improved accuracy, a smaller window size that matches the kernel size of the CNN would likely yield better results. This is because a smaller window size could provide a more detailed view of the signal, allowing for more precise identification of the characteristic features used for classification.

E. GUI Interface

The below figure shows the GUI interface for uploading the file containing the ADC dataset. It then processes the file and displays the predicted fields along with the plots.

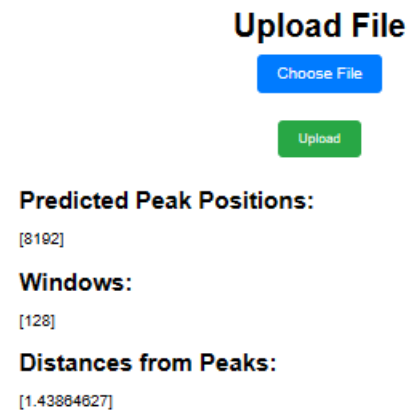

Fig.24. GUI Interface for uploading the files and displaying the results.

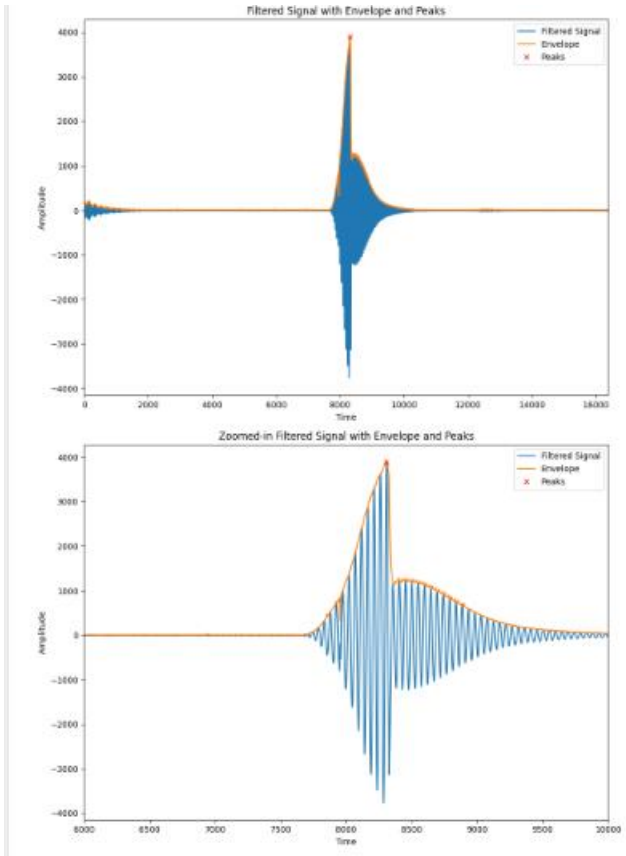

Fig.25. GUI Interface for displaying the graphs

V. CONCLUSION & FUTURE SCOPE

The aim of this project was to ascertain the position of the first echo from ultrasonic waves, utilizing the Red Pitaya STEMLAB measurement board and an Ultrasonic sensor SRF02. We have found that ultrasonic waves exhibit varied behaviour when interacting with different materials, as evidenced by distinct patterns in the reflected signals captured by the sensor. The Red Pitaya board has been instrumental in data collection, allowing us to discern these patterns clearly.

The classification behaviour observed through our experiments may show significant variances, especially when there are notable changes in the experimental setup. Through a comparative analysis, we learned that each model has its unique strengths in handling the data from the 1-meter ultrasonic sensor measurements.

The comparative analysis of CNN, Random Forest, and XGBoost models on the 1m dataset has provided valuable insights into the strengths and weaknesses of each approach. The CNN model emerged as the top performer, likely due to its ability to harness spatial dependencies within the data—a characteristic that is particularly relevant for signal processing tasks. The Random Forest model also demonstrated commendable accuracy, suggesting that ensemble methods are robust and effective for this class of problems. XGBoost, while slightly trailing, still showed a respectable level of accuracy and remains a competitive option, especially considering its speed and scalability.

In essence, while the CNN model showcased a high degree of accuracy in classifying hard objects at 1 meter, the expanded application to a mixed dataset demonstrates a respectable level of accuracy, given the increased complexity. This outcome supports the model's use in practical applications, highlighting its strengths and areas for potential refinement to enhance its discriminative capabilities further.

The impact of window size has highlighted the importance of meticulous parameter tuning in signal processing and machine learning, emphasizing the necessity to balance accuracy with computational efficiency and the intrinsic characteristics of the input signals.

Despite not fully understanding the complexities within the data features, the classification models have significantly advanced our research. A clear path to improving classification rates is to augment the training data volume. Training larger, pre-trained models with our experimental data could facilitate better prediction of precision of the first echo.

Future Scope:

For future research to bolster the models' robustness and practical applicability, several initiatives could be undertaken:

Model Optimization: Refining model hyperparameters with advanced techniques and experimenting with intricate CNN architectures or enhanced ensemble methods, could optimize performance.

Feature Engineering: Creating sophisticated features to better capture signal characteristics may improve the models' discriminatory power.

Class Imbalance: Techniques like SMOTE or adaptive sampling to counter class imbalance can be crucial in improving underrepresented class predictions.

Real-time Processing: Modifying models for real-time analysis can be pivotal for applications requiring instant decision-making.

Model Ensemble: Leveraging multiple models in an ensemble approach could improve performance by capitalizing on the strengths of individual models.

Interpretability and Explainability: Implementing frameworks to understand model decisions will be critical in scenarios where justification is crucial.

Transfer Learning: Fine-tuning CNNs with transfer learning techniques might yield superior results, especially when pre-trained on extensive datasets.

Hardware Optimization: Tailoring models for specific hardware can ensure their deployment readiness for real-world applications.

Extended Validation: Evaluating models against an independent validation set would offer a more robust assessment of their predictive prowess.

Through these strategic research directions, the potential for enhancing not only the predictive accuracy but also the utility and trustworthiness of AI systems in operational settings is vast. The iterative process of improvement is key to the development of reliable and high-performance AI-driven systems.

VI. ACKNOWLEDGMENT

We would like to express our gratitude to Prof. Dr. Andreas Pech and Prof. Dr. Peter Nauth for giving us the chance to work on this subject under their supervision and for giving us the right direction to finish the required project and this report. The lectures on "Autonomous Intelligence System" and "Machine Learning" were very beneficial in providing the background knowledge and motivation for writing the report for the seminar. We also want to express our gratitude for Prof. Julian Umansky important contribution in providing all the tools necessary for the task to be completed successfully. If we do not express our gratitude to the writers of the references and other works cited in this work, we will be in breach of our responsibility.

VII. REFERENCES

[1] Richards, Mike, Pi Updates and Red Pitaya, Radio User. Bournemouth, UK: PW Publishing Ltd. 11, 2016.

[2] R. C. Luo, S. L. Lee, Y. C. Wen and C. H. Hsu, "Modular ROS Based Autonomous Mobile Industrial Robot System for Automated Intelligent Manufacturing Applications," 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), [Online]. Available: doi: 10.1109/AIM43001.2020.9158800..

[3] "SRF02 Ultrasonic range finder," robot-electronics, [Online]. Available: https://www.robot-electronics.co.uk/htm/srf02tech.htm.

[4] P. N. T. Wells, "Ultrasonic attenuation measurements in liquids and solids," *Physical Acoustics,* vol. 3, pp. 219-294, 1966.

[5] S. G. Kim, "Ultrasonic reflection and refraction at material boundaries," *Journal of the Acoustical Society of America,* vol. 39, pp. 600-607, 1966.

[6] M. P. B. M. A. R. J. H. Arnold, "Multipath effects in ultrasonic waveforms," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 38, pp. 314-323, 1991.

[7] A. H. a. N. P. M. a. M. R. Pech, "A new Approach for Pedestrian Detection in Vehicles by Ultrasonic Signal Analysis," pp. 1-5, 2019.

[8] Z. F. H. e. al., "A low-cost ultrasonic distance measurement system," *IEEE* , 2013 .

[9] A. J. H. a. P. M. Grant, "Precision distance measurement using pulsed ultrasonics," 1993.

[10] D. J. N. a. A. D. S. Fitt, "Distance measurement using ultrasound," *Journal of Physics E: Scientific Instruments,,* 1982.

[11] K. F. W., Hilbert Transform, Vol. 2, Cambridge, UK: Cambridge University Press, 2009.

[12] B. R., The Fourier Transform and it`s application, McGraw-Hill, 1965.

[13] K. J. Piczak, "ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS," *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP),* p. 6, 2015.

[14] Z. C. L. R. W. John R. Hershey, "Deep clustering: Discriminative embeddings for segmentation and separation," 2015.

[15] I. S. G. E. H. A. Krizhevsky, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, Nevada, 2012.

[16] Y. B. a. G. H. Y. LeCun, "Deep learning," *Nature,* vol. 521, pp. 436-444, 2015.

[17] L. Breiman, "Random forests," *Machine Learning,* vol. 45, pp. 5-32, 2001.

[18] T. K. Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, Canada, 1995.

[19] T. C. a. C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

[20] A. L. a. M. Wiener, "Classification and regression by randomForest," *R News,* vol. 2, pp. 18-22, 2002.

[21] P. S. a. M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry,* vol. 36, pp. 1627-1639, 1964.

[22] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, pp. 674-693, 1989.