Master of Engineering
Information Technology
Machine Learning
Prof. Dr. Andreas Pech

Reliability test and improvement of a sensor system for object detection

Master of Engineering
Information technology
Deblina Karmakar
1427365
deblina.karmakar@stud.fra-uas.de

Master of Engineering
Information Technology
Indranil Saha
1427190
indranil.saha@stud.fra-uas.de

Abstract- This comprehensive report explores the forefront of enhancing echo detection accuracy through the innovative application of Frequency-Inversion Ultrasound Signals (FIUS), especially pertinent in environments where the limitations of traditional optical sensors, such as poor visibility or physical blockades, necessitate alternative sensing modalities. The utilization of the Red Pitaya platform has facilitated the integration of advanced machine learning models—including, but not limited to, Convolutional Neural Networks (CNN), Random Forest, and XGBoost—coupled with elaborate data processing techniques, all aimed at significantly bolstering the performance and dependability of sensor systems. The pivotal challenge of echo detection, an essential mechanism for the precise measurement of object distances, which is vital for the seamless operation and efficiency of sensor systems. A thorough investigation has been carried out, focusing on the signal's intensity and its spectral characteristics over a diverse range of distances and angles, with the ultimate goal of crafting a framework to refine and optimize echo detection methodologies. Strategies such as the application of confusion matrices and the implementation of rigorous error management protocols have been adopted to systematically reduce the uncertainties inherent in echo detection processes. The study also examines the potential of machine learning algorithms to adaptively filter and process FIUS data, enhancing signal-to-noise ratios and improving the accuracy of distance estimations. By harnessing the power of machine learning for the intelligent analysis and interpretation of sensor data, this work lays the groundwork for significant enhancements in the capability and versatility of intelligent systems. The implications of these advancements are far-reaching, promising to elevate the operational intelligence and autonomy of systems across a myriad of applications, from autonomous vehicles and robotics to surveillance and environmental monitoring.

Keywords- FIUS, Red Pitaya, data processing, Machine Learning, Convolutional Neural Networks, Random Forest, Confusion Matrix.

I. INTRODUCTION

The advent of Intelligent Systems marks a significant leap towards achieving operational autonomy in various sectors including robotics, automotive, and unmanned aerial vehicles. At the heart of these systems lies the critical role of sensing technologies that enable it to perceive their environment, make informed decisions, and navigate with minimal human intervention. Among the myriad sensing techniques, echo detection from Frequency- Inversion Ultrasound Signals (FIUS) emerges as a crucial technology, especially in scenarios where traditional optical sensing methods fall short due to poor visibility, obstructions, or environmental conditions. This project focuses on advancing the reliability of echo detection using FIUS, a cornerstone upon which the efficacy and decision-making capabilities of multiple systems significantly depend.

Echo detection involves the creation of an acoustic signal, and detecting the acoustic echo reflected back. One common application of echo detection is object and position detection. For instance, advanced vehicles often use echo detection to obtain information regarding positioning of objects external to the vehicle[1]. On the other hand, ultrasonic technology develops faster and faster, more and more widely used, but sometimes, because of the reason the echo detection circuit accuracy is not high, detection technology is not good enough, leading to inability to extract the desired information from the echo [2].

The principle of echo detection, though straightforward in theory, encompasses a multitude of challenges when applied in complex, real-world environments. Factors such as signal interference, noise, and the variability of object surfaces can greatly affect the accuracy of echo-based distance measurements and object identification. To address these challenges, this project delves into sophisticated methodologies for enhancing the reliability of echo detection. It examines the intricacies of error propagation in signal processing and its implications on the precision of the measurements obtained. Understanding and minimizing these errors is vital for improving the robustness and

dependability of AIS in executing tasks that require high levels of precision.

Furthermore, the project explores the development of comprehensive models utilizing different algorithms to characterize the behavior of ultrasonic signal intensity and spectra as they propagate through space, interacting with various objects at different distances and angles.

II. METHODOLOGY

In order to conduct the experiment, a detailed and thorough theoretical background study of the involved components and algorithms has been conducted. This particular section of the paper covers in detail the theoretical approach, procedures, and strategies of the implemented project work.

A. Red Pitaya Measurement Board

The Red Pitaya STEM Lab is a versatile test and measurement platform built on a system-on-a-chip (SoC) [3] originally developed by Xilinx. It is a compact, versatile, and powerful test and measurement device that serves as an essential tool for a wide range of users in the fields of science, technology, engineering, and mathematics (STEM). Built on a robust SoC architecture, the STEM Lab is designed to function as multiple devices in one, including but not limited to an oscilloscope, signal generator, spectrum analyzer, and frequency analyzer. The specifications of the Red Pitaya STEM lab can be described as follows –

- SoC(System-on-a-Chip): Integrates FPGA technology, offering customizable and powerful processing capabilities suitable for various computational tasks.
- Analog and Digital I/O: Equipped with 2 analog RF inputs and outputs, supports a wide frequency range, alongside numerous digital I/O ports for versatile interfacing.
- Connectivity: Provides Ethernet and Wi-Fi options for remote operation and data analysis, along with USB connections for direct interfacing and programming.
- Storage and Expansion: Features a micro-SD card slot for additional data storage and software, as well as an extension connector for modular expansions and custom projects.
- Software Compatibility: Runs on an open-source Linux operating system, facilitating the development of custom applications and the use of a broad spectrum of software tools and libraries.

B. Ultrasonic Sensor

The red Pitaya platform, mentioned in the previous section, is paired with the Ultrasonic Sensor SRF02 [4], a compact PCB-mounted ultrasonic rangefinder featuring a single transducer design. This design choice results in a minimum detection range of about 15 cm (6 inches), which is larger than that of dual transducer models due to its combined use for both signal emission and reception.

With a 5V power supply, the sensor is mounted on a stand and is designed for seamless wireless data transmission to a laptop for further analysis, facilitated by the Red Pitaya's capabilities. Running on the GNU/Linux operating system, the sensor's data can be easily managed and analyzed on both computers and mobile devices. The Primary unit of the Red Pitaya consists of the below specifications –

- 16 standard digital input/output ports
- 2 analog RF input/output ports
- 1 micro-SD card slot for storage
- 1 RJ45 socket for Ethernet connectivity
- 1 USB port and 1 micro-USB port for console connections.

Capable of handling RF signals, the Red Pitaya operates within a 50MHz frequency range, enabling it to send and receive RF signals effectively. Figure 1 below shows the ultrasonic sensor and red pitaya device, that has been utilized in this project work.

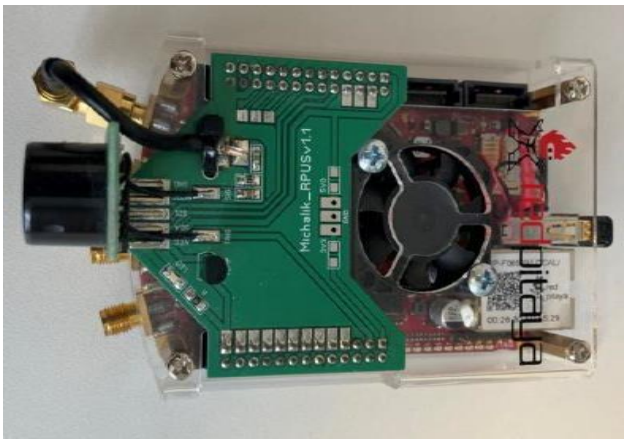

Figure 1: Ultrasonic sensor and red pitaya device

On the other hand, the pin functions of the sensor are displayed vividly in figure 2.

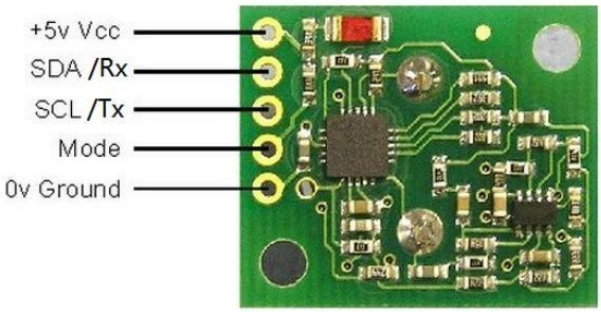

Figure 2: Pin function of the SRF02 Ultrasonic sensor

C. Fast Fourier Transform

The fast Fourier transform is a computational tool which facilitates signal analysis such as power spectrum analysis and filter simulation by means of digital computers. It is a method for efficiently computing the discrete Fourier transform of a series of data samples (referred to as a time series)[5]. By far, the most commonly used FFT is the Cooley-Tukey algorithm. This is a divide-and-conquer algorithm, recursively breaking down a DFT of any composite size n=n1*n2, into smaller DFTs of sizes n1 and

n2, along with O(n) multiplications by complex roots of unity traditionally called twiddle factors[6]. Below formula depicts the equation of DFT -

$$X_k = \sum_{m=0}^{n-1} x_m e^{-i2\pi km/n}$$

Where k=0,1,2,..,n-1 and $e^{i2\pi/n}$ is a primitive n'th root of 1.

D. Confusion Matrix

A confusion matrix is a performance evaluation tool in machine learning, representing the accuracy of a classification model. It displays the number of true positives, true negatives, false positives, and false negatives. This matrix aids in analyzing model performance, identifying misclassifications, and improving predictive accuracy. It is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives a holistic view of how well the classification model is performing and what kinds of errors it is making[7]. Figure 3 showcases a 2X2 matrix for a binary classification problem.

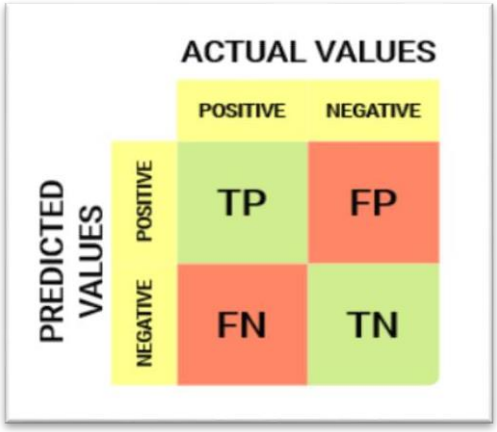

Figure 3: Confusion Matrix

The important terms in a confusion matrix are described as below –

True Positive (TP):

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was positive, and the model predicted a positive value.

True Negative (TN):

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was negative, and the model predicted a negative value.

False Positive (FP) – Type I Error

- The predicted value was falsely predicted.
- The actual value was negative, but the model predicted a positive value.
- Also known as the Type I error.

False Negative (FN) – Type II Error

- The predicted value was falsely predicted.
- The actual value was positive, but the model predicted a negative value.
- Also known as the Type II error.

From the confusion matrix, several key metrics can be derived to assess the effectiveness of a classification algorithm, as outlined below:

Accuracy: This metric calculates the proportion of correctly identified samples out of the total dataset. It reflects the classifier's overall correctness. The formula for accuracy is given by:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision: Precision represents the fraction of true positive predictions out of all positive predictions made by the model. It indicates the accuracy of positive predictions. The precision formula is as follows –

Precision = TP / (TP+ FP)

Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive samples. It assesses the model's ability to identify all relevant instances. The recall formula is as follows –

Recall = TP / (TP + FN)

Specificity: Specificity calculates the fraction of true negative predictions out of all actual negative samples, showing the model's accuracy in identifying negatives. The formula for specificity is as follows –

Specificity = TN / (TN + FP)

F1 Score: The F1 score is the harmonic mean of precision and recall, serving as a balanced measure when both precision and recall are crucial. Its formula is –

F1 Score = 2*(Precision*Recall)/(Precision+Recall)

E. Model Training

A branch of AI, Machine Learning (ML) focuses on creating algorithms and models allowing computers to learn from data and autonomously make decisions or predictions, without being explicitly programmed. It emphasizes enabling computers to gain knowledge from experiences, moving away from fixed rules and human intervention.

In machine learning, models are trained on datasets with input and corresponding output values to create functions that predict outputs for new, unseen data. This involves iteratively refining model parameters to reduce discrepancies between predicted and actual outputs.

The realm of machine learning encompasses various algorithmic approaches, such as supervised learning, unsupervised learning, and reinforcement learning. Within supervised learning, algorithms undergo training on a dataset that is labeled, meaning each piece of input data is matched with the correct output value. Conversely, unsupervised learning involves training algorithms on datasets that are not labeled, aiming to uncover hidden patterns and structures

within the data. For the purpose of this experiment, supervised learning is employed to construct different distinctive models utilizing the Convolutional Neural Network (CNN), Random Forest and XGBoost.

F. Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections[8][9]. Below figure 4 displays a typical CNN architecture consisting of different layers.

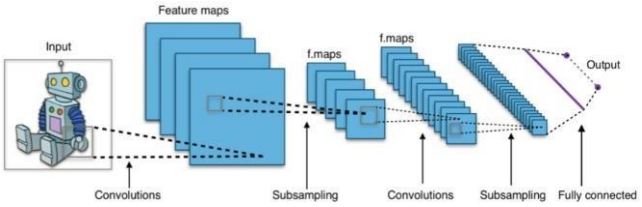

Figure 4: A typical CNN architecture

The working of the CNN model can be described as per the following points –

- Convolutional Neural Networks (CNNs) are composed of layers containing artificial neurons or nodes, which perform calculations to produce activation maps, identifying key features in input data, such as images.
- Each node calculates a weighted sum of its inputs and applies an activation function to generate an output, focusing on specific features like edges or textures in the initial layers.
- Through a process known as convolution, these networks can identify and emphasize important visual features by applying filters that detect patterns, edges, and other significant elements.
- As data progresses through the layers, CNNs use max pooling to reduce dimensionality and focus on the most relevant features, ensuring that subsequent layer's work with the most informative aspects of the data.
- The final layers of a CNN typically involve classification, where the network makes predictions or identifications based on the processed features, assigning input data to predefined categories.
- Training a CNN involves adjusting the weights of the nodes based on the accuracy of the output compared to known labels in a training dataset, with careful attention to avoid overfitting, ensuring the model generalizes well to new, unseen data [10].

G. Random Forest

A Random Forest is like a group decision-making team in machine learning. It combines the opinions of many "trees" (individual models) to make better predictions, creating a more robust and accurate overall model. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. The steps involved in the algorithm of Random Forest are as follows –

- In the Random Forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.
- Individual decision trees are constructed for each sample.
- Each decision tree will generate an output.
- Final output is considered based on Majority Voting or averaging for Classification and regression, respectively [11].

H. XGBoost

XGBoost is a machine learning algorithm that belongs to the ensemble learning category, specifically the gradient boosting framework. It utilizes decision trees as base learners and employs regularization techniques to enhance model generalization. Known for its computational efficiency, feature importance analysis, and handling of missing values, XGBoost is widely used for tasks such as regression, classification, and ranking.

In contrast to bagging techniques like Random Forest, in which trees are grown to their maximum extent, boosting makes use of trees with fewer splits. Such small trees, which are not very deep, are highly interpretable. Parameters like the number of trees or iterations, the rate at which the gradient boosting learns, and the depth of the tree, could be optimally selected through validation techniques like k-fold cross validation.

Among the various ensemble techniques, the gradient boosting consists of majorly three steps –

- An initial model F0 is defined to predict the target variable y. This model will be associated with a residual (y-F0).
- A new model h1 is fit to the residuals from the previous step.
- Now, F0 and h1 are combined to give F1, the boosted version of F0. The mean squared error from F1 will be lower than that from F0.

$$F_1(x) \leftarrow F_0(x) + h_1(x)$$

Performance of F1 is improved, by modelling the residuals of F1 and creating a new model F2.

$$F_2(x) \leftarrow F_1(x) + h_2(x)$$

This can be done for 'm' iterations, until residuals have been minimized as much as possible.

$$F_m(x) \leftarrow F_{m-1}(x) + h_m(x)$$

In this method, the additive learners do not disturb the functions created in the previous steps. Instead, they impart information of their own to bring down the errors.

III. Implementation

A. *Implementation Workflow*

The implementation is outlined using a flowchart, as shown in figure 5, for a machine learning model starting with the collection, and reading of data. This data then undergoes a series of transformations including cleaning and normalization, preparing it for further analysis. Key stages highlighted include the identification of significant data points through peak analysis and the subsequent organization and labeling of data, crucial steps for pattern recognition and categorization.

Following the data preparation, the process involves defining the model tailored to the data's characteristics and the analytical goals. This model is trained with the organized data, learning to identify patterns and make predictions. Upon reaching satisfactory performance levels, the model is saved, allowing for future reuse or deployment, thus encapsulating a streamlined approach from data collection to model readiness for predictive tasks.

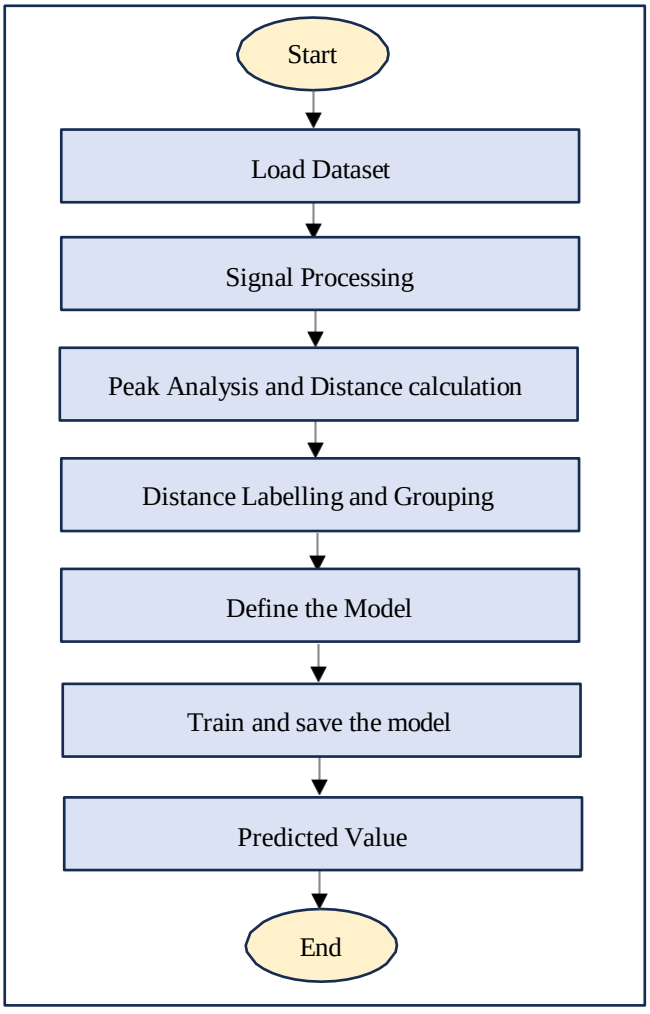

Figure 5: Flowchart of the implemented workflow

B. *Measurement Setup and data collection*

A total of 1000 ADC data points are collected for each object type across five experimental runs. Measurements have been taken using the FIUS sensor setup and manually verified with a standard measuring scale to ensure precision. The data encompasses a wide range of distances, environmental conditions, and object materials to simulate varied real-world scenarios.

In the Robolab of Frankfurt University of Applied Sciences, the FIUS sensor is positioned at the top on an elongated black metal stand as projected in figure 6a. The object under consideration is situated directly beneath the FIUS sensor on a short white stand, as depicted in figure 6b.

Figure 6a) : Manual measurement of the distance from the FIUS sesnor to the ground using a folding meter stick

Figure 6b) : Manual measurement of the distance of the hard object to the ground using a folding meter stick

On the other hand, the process of data measurement is visually represented within the measurement software's Graphical User Interface (GUI). This GUI has been already pre-configured to specifically analyze analog data collected from ultrasonic sensors. Upon completion of the measurement setup, the software is adeptly prepared to not only capture but also store the raw data effectively. This is a critical step, ensuring that all data collected is accurately represented and readily available for further analysis. Figure 7 offers a concise glimpse into this process, showcasing the snippet of the software as it captures and handles the data.

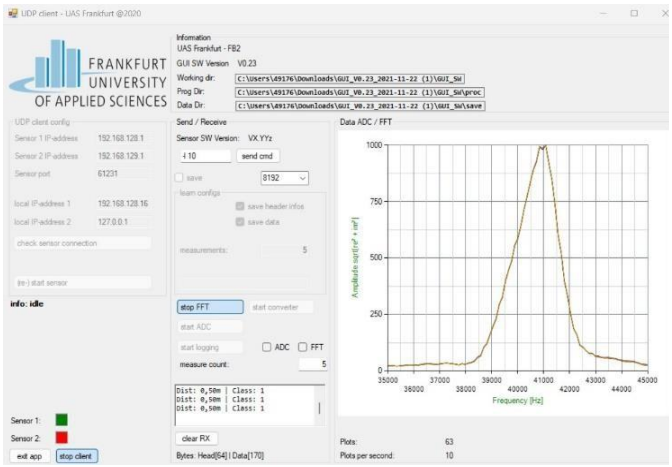

Figure 7: Data capture from the measurement software

For the measurement purpose, text-based ADC data format has been used that has been exported from the software. A snippet of the raw data has been shown in figure 8. Each data has 16834 columns overall, with rows representing amplitude values in the range of 0 to 1000 and columns representing frequency values.

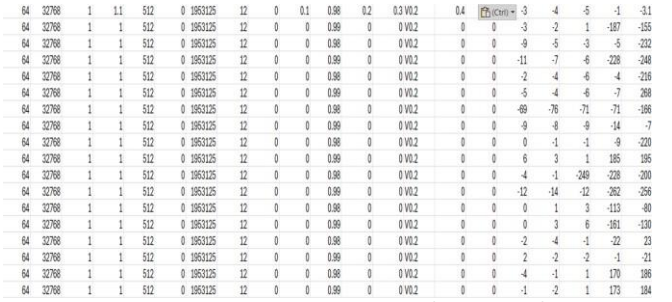
Figure 8: Raw measurement data from the software

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.

C. Pre-Processing of the obtained data

Pre-Processing of the raw data from the ultrasonic sensor is done in several stages. They are mentioned as follows-

Extraction of the relevant information from the obtained dataset: The '*read_and_prepare_data*' function plays a pivotal role in the data processing pipeline by loading signal data from a CSV file, with a particular focus on columns that contain pertinent information. This initial step is crucial for the preparation of raw ultrasonic signals, setting the stage for in-depth analysis. It involves filtering and extracting the most relevant data from the multitude of available information, ensuring that only the data of interest is forwarded for subsequent processing. This targeted approach helps streamline the analysis, making it more efficient and focused on the signals that are most likely to yield valuable insights.

Signal Windowing: The '*apply_window*' function serves as a critical step in signal processing, specifically designed to address and mitigate spectral leakage—a common issue that can distort frequency analysis. By applying a Hanning window to each signal, this process effectively shapes the data, tapering the beginning and end of each signal. This tapering is essential for smoothing transitions and minimizing abrupt changes, which, in turn, significantly enhances the accuracy and reliability of frequency analysis. The level of precision is crucial for the accurate detection of echoes, a fundamental aspect of analyzing ultrasonic signals, ensuring that the data analysis leads to trustworthy and actionable insights.

Noise Reduction and Peak Detection: In the '*reduce_noise_and_label*' process, the initial step involves converting each signal from its time domain to the frequency domain through the Fast Fourier Transform (FFT). This transformation is pivotal for identifying and isolating the signal's frequency components. Following this, a Power Spectral Density (PSD) threshold is applied as a filter to effectively eliminate background noise, sharpening the focus on relevant signal components. The process then employs an inverse FFT to revert the filtered signal back to the time domain, but with significantly reduced noise. To further refine the signal for analysis, the Hilbert Transform is applied to the denoised signal to extract its envelope. This envelope is crucial for the effective detection of peaks using the '*find_peaks*' function, enabling more precise identification of significant features within the signal.

Labelling the data: This specialized function is adept at taking a compilation of peak positions identified within signal data and systematically arranging these observations into an organized structure. This structure not only highlights the occurrence of peaks but also delineates their presence within specific temporal segments or windows across a multitude of signals. The function embarks on this task by first ascertaining the total number of distinguishable windows present within a signal. This determination is made by dividing the entire length of the signal by the predefined width attributed to each window, thereby establishing a count for the possible windows.

Following the calculation of windows, the function proceeds to construct a matrix, referred to as 'y_label'. This matrix is designed to encapsulate binary labels that correspond to each signal across the entirety of the windows, effectively indicating the presence or absence of peaks within each window. The dimensions of this matrix are carefully defined, with the number of rows equaling the total number of signals—this is inferred from the length of the list containing peak positions (peaks_list). Concurrently, the number of columns within the matrix matches the previously calculated number of windows (n_windows). Through this process, the function achieves a comprehensive overview, mapping out the distribution and incidence of peaks across various signals and temporal windows, thereby facilitating a structured analysis of signal characteristics.

The described function meticulously processes a list of detected peak positions within signal data, structuring these observations in a way that clearly marks the occurrence of peaks across distinct time frames within multiple signals. To achieve this, it begins by calculating the total number of

potential time windows for any given signal, a figure derived by dividing the signal's total duration by the designated width of each time window.

For every individual signal under analysis, the function then scrutinizes each identified peak position. When a peak is found (signified by a position value that is not negative), it calculates the specific time window that this peak corresponds to by dividing the peak's position by the width of the window. This calculation determines exactly where within the temporal structure of the signal the peak occurs.

Upon determining the correct time window for a peak, the function updates a specially designed matrix, known as the label matrix, setting the element that corresponds to this signal and window to 1. This action signals the presence of a peak within that precise window, creating a binary map of peak occurrences. Conversely, time windows that do not house any peaks are marked with a 0, clearly indicating a lack of peak activity in those intervals.

D. Implementation of CNN

The function is designed for the training of a Convolutional Neural Network (CNN), which is adept at handling data characterized by spatial relationships, such as images or time-series datasets. The process unfolds in several meticulously structured steps:

Initialization Phase: Training parameters are established, encompassing the verbosity level, the total number of training epochs, and the specified batch size.

Model Definition:

- A linear sequence of layers is constructed using '*Sequential()*' function.
- For convolution operations on one-dimensional data, *Conv1D* layers are integrated, each configured with 64 filters and a kernel size of 3.
- To mitigate the risk of overfitting, a *Dropout* layer is applied, setting the rate at 0.5.
- Dimensionality reduction is achieved through *MaxPooling1D*, enhancing the model's ability to generalize.
- A *Flatten* layer is utilized to transform the 2D feature maps into a 1D vector, preparing the data for dense layer processing.
- The architecture includes *Dense* layers, which are fully connected, with the final layer employing softmax activation to calculate probabilities across various classes.

Model Compilation: In this step, the model undergoes compilation with the categorical cross entropy loss function and the Adam optimizer, setting the groundwork for training.

Training: The model is subjected to training using the provided datasets (x_train, y_train), adhering to the previously set epochs and batch size.

Evaluation: Post-training, the model's performance and accuracy are rigorously evaluated against the test dataset.

Model Saving: Subsequent to evaluation, the fully trained model is preserved on the file system, facilitating future access and utilization.

Return: Ultimately, the function concludes by returning the trained model, ready for deployment.

E. Implementation of Random Forest

The Random Forest algorithm is a powerful ensemble learning technique known for its versatility and ease of use. It builds upon the concept of decision trees, aggregating multiple such trees to form a "forest" that enhances the predictive accuracy and controls over-fitting. The approach towards deploying a Random Forest model involves several critical stages, each contributing to the model's overall effectiveness:

Parameter Grid Construction: The first step in optimizing a Random Forest model involves setting up a comprehensive grid of hyperparameters. This grid encompasses a variety of parameters crucial for the model's performance, including the number of trees in the forest (*n_estimators*), the maximum number of features considered for splitting a node (*max_features*), the maximum depth of each tree (*max_depth*), the minimum number of samples required to split a node (*min_samples_split*), the minimum number of samples necessary at a leaf node (*min_samples_leaf*), and the decision to use bootstrap samples for building trees (*bootstrap*). This grid serves as the foundation for the subsequent search for the optimal set of parameters.

Randomized Hyperparameter Search: To efficiently navigate the vast parameter space outlined by the grid, a 'RandomizedSearchCV' is employed. This method diverges from exhaustive search techniques by randomly sampling from the parameter space, offering a balance between exploration and resource consumption. The output of this phase is the identification of the best hyperparameter combination, crucial for constructing the most effective Random Forest model.

Model Training with Best Parameters: Equipped with the optimal set of parameters discovered in the previous step, a new Random Forest model is instantiated. This model is then trained on the entire training dataset, leveraging the best hyperparameter values to ensure superior learning and generalization capabilities.

Model Preservation: Upon successful training, the refined model is preserved onto the file system, typically in a pickle file format. This step ensures that the model can be easily retrieved and deployed for future predictions without the need for retraining.

Prediction and Performance Evaluation: The trained model is then put to the test, predicting outcomes for both the training and test datasets. The effectiveness of these predictions is quantified using the F1 score, a balanced metric considering both precision and recall, for each set of predictions. These scores are crucial indicators of the model's

performance and its ability to generalize beyond the training data.

Return of the Trained Model: The culmination of this process is the return of the trained Random Forest model, now fine-tuned with the best possible parameters. This model stands ready for application in predictive tasks, equipped with the ability to provide high-quality predictions.

F. Implementation of XGBoost

XGBoost stands as a prominent and efficient gradient boosting library that has garnered acclaim for its performance on structured or tabular data across various machine learning competitions and applications. Implementing an XGBoost model involves a series of meticulously planned steps, each designed to maximize the model's predictive power and efficiency:

Model Initialization: The process starts with the initialization of an XGBoost classifier. This step involves configuring a set of hyperparameters that define the model's structure and learning process. Key parameters include the objective function, which is often set to *binary:logistic* for binary classification tasks, indicating the model's goal to predict a binary outcome. Additionally, the number of estimators specifies how many boosting rounds or trees the model shall be built. The maximum depth of trees controls the complexity of the model, the learning rate dictates the step size at each iteration to prevent overfitting, and the subsample ratio helps in reducing variance by training on a subset of the data at each step.

Model Training: With the classifier initialized, the model is then trained by fitting it to the training data. This process involves iteratively building trees, each designed to correct the errors of its predecessors, thereby continuously improving the model's accuracy with each round. The training phase is crucial, as it allows the model to learn from the structure and patterns within the data, adjusting its parameters to minimize prediction errors.

Model Saving: After the model has been trained and its parameters optimized, it is saved to the file system, typically in a pickle file. This serialization step ensures that the model can be stored and later retrieved without the need to retrain it from scratch, facilitating easy deployment in predictive applications or further analysis.

Evaluation: The model's effectiveness is then evaluated by making predictions on both the training and test datasets. The F1 score, which serves as the evaluation metric, is calculated for each set of predictions. The F1 score, being the harmonic mean of precision and recall, provides a more balanced measure of a model's performance, especially in cases of imbalanced datasets where accuracy alone might be misleading.

Return of the Trained Model: The final step is the return of the trained XGBoost model. This model, now ready and equipped with the learned parameters, stands as a powerful tool for making predictions on new data, embodying the culmination of the training and optimization efforts.

IV. RESULT AND DISCUSSION

This section deals with result and observations made at various stages of the project work. At first, it deals with how the raw data, extracted from the sensor, is processed into FFT data for further analysis. Afterwards, it displays the results obtained from different models and provides a comparative study among them.

In the initial phase of this project work, data acquisition is conducted using an ADC to capture readings from both hard surfaces and persons positioned at distances of 1 meter and 50 centimeters respectively. To ensure robust data sampling, 1000 measurements were collected for each combination of distance and surface condition.

Following data acquisition, a custom software algorithm has been developed to process the acquired ADC data. This algorithm performs Fast Fourier Transform (FFT) analysis on the raw ADC readings, thereby converting them into frequency domain representations, enabling a deeper understanding of the characteristics embedded within the measured signals. Figure 9 and figure 10 showcases the ADC to FFT plot for hard object and person at 1m distance from the sensor.

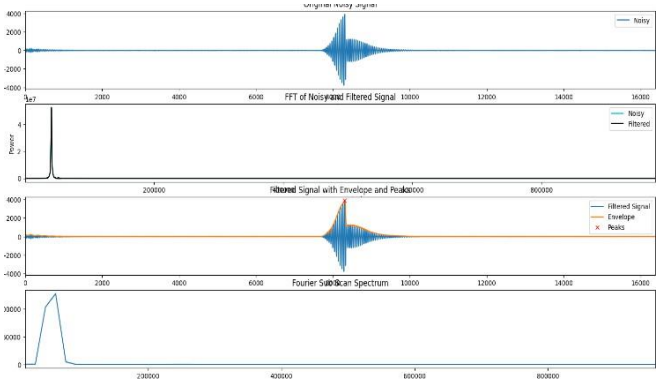

Figure 9: ADC to FFT plot for hard object placed at 1m

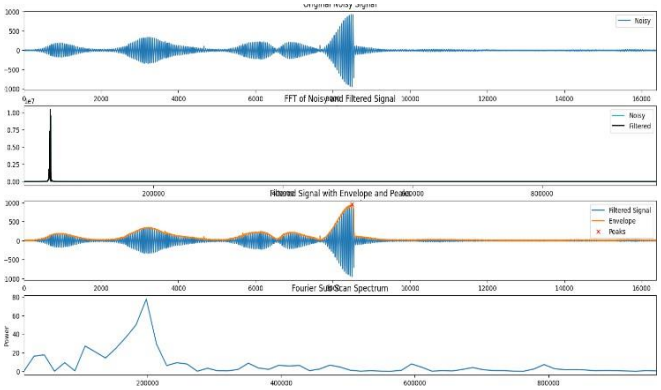

Figure 10: ADC to FFT plot for person sitting at 1m

Similar to the above case, position of both the hard object and the person is changed as displayed in figure 11 and 12 and again raw data is collected by the sensor in order to convert it into FFT data for further analysis and clarifications.

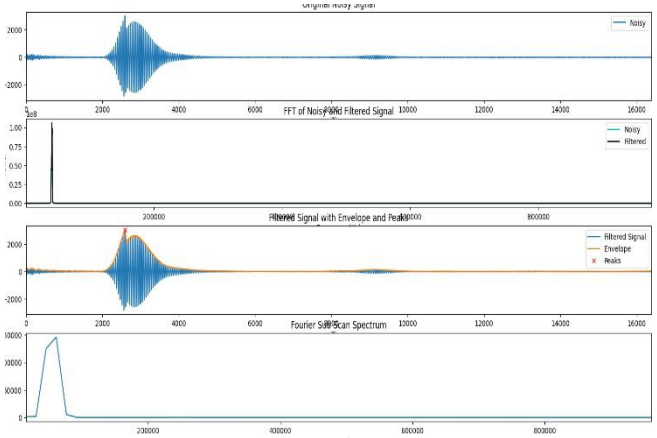

Figure 11: ADC to FFT plot for hard object placed at 50cm

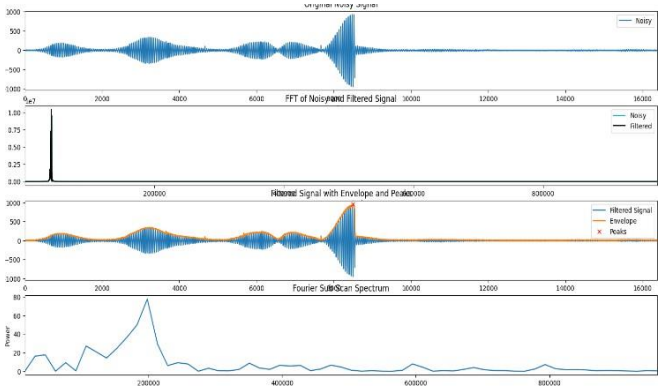

Figure 12: ADC to FFT plot for person placed at 50cm

Overall, this approach enables significantly more refined insights into the properties of both hard objects and person, by leveraging FFT analysis to extract frequency domain information.

The code complements this methodology by serving as a tool for processing and visualizing the ultrasonic measurement data stored in a CSV file. Upon importing necessary libraries, including NumPy, Pandas, Matplotlib, and SciPy, the script loads the dataset and selects specific set of columns for analysis. Each signal within the dataset undergoes a series of signal processing procedures, including Fourier Transform computation, noise filtering, and envelope detection using the Hilbert Transform. Additionally, peaks are detected in the envelope of the filtered signal using SciPy's *find_peaks* function.

Furthermore, the script segments the signal into windows, applies a Hanning window to each segment, and computes the FFT to analyze frequency content. This results in the generation of four distinct plots for each signal, showcasing various aspects such as the original noisy signal, FFT analysis of both noisy and filtered signals, filtered signal with envelope and detected peaks, and the Fourier Sub Scan Spectrum. Finally, the generated plots are saved as PNG files for further analysis or visualization.

In summary, the script provides a means of gaining insights into the characteristics of ultrasonic signals and facilitates interpretation of the underlying data through visual representation, complementing the methodology outlined in the study.

Upon training the three distinct models on Analog-to-Digital Converter (ADC) data, specifically targeting hard objects placed at a distance of 1 meter, we observed a variety of results. These outcomes provide insights into the models' performance and effectiveness in interpreting and analyzing the ADC data under these particular conditions. Each model, leveraging its unique approach to data analysis and prediction, contributed to a comprehensive understanding of how different algorithms can be applied to ultrasonic measurement data for objects at this specified range.

A. Convolution Neural Network (CNN)

The Convolutional Neural Network (CNN) model demonstrates exceptional performance in precision, recall, and F1-score, particularly for the predominant classes (128, 129, 130), showcasing its proficiency in accurately predicting these categories. It achieves an overall accuracy rate of 90.5%, indicating that a significant majority of predictions are correct. Furthermore, the model maintains high weighted averages for both the F1-score and recall, approximately 0.91, indicating robust performance across various classes. This performance considers the support for each class, reflecting the model's effectiveness in handling different class instances uniformly well. Figure 13 shows an overall model summary while figure 14 shows the classification report of the CNN model.

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv1d (Conv1D)                 │ (None, 16382, 64)      │           256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv1d_1 (Conv1D)               │ (None, 16380, 64)      │        12,352 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 16380, 64)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling1d (MaxPooling1D)    │ (None, 3276, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 209664)         │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 1052)           │   220,567,580 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 256)            │       269,568 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 220,849,758 (842.47 MB)
 Trainable params: 220,849,756 (842.47 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2 (12.00 B)
```

Figure 13: CNN model summary

```
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
```

Figure 14: Classification report of the CNN model

B. Random Forest

Upon optimizing hyperparameters, the Random Forest model exhibited its optimal performance configured with 100 trees (*n_estimators*), a criterion of a minimum of 5 samples to split an internal node (*min_samples_split*), a requirement of just 1 sample at a leaf node (*min_samples_leaf*), and max_features determined by the square root of the total number of features. The trees were allowed a maximum depth (*max_depth*) of 30, with bootstrapping explicitly disabled (bootstrap: set to False). While this model demonstrated commendable accuracy across predominant classes, it showed a marginally lower efficacy for class 130 when contrasted with the CNN model's performance. Notably, this model surpassed in overall accuracy, achieving a rate of 92.5% for correct predictions. Additionally, the model maintained high and consistent weighted averages for both the F1-score and recall, approximately 0.92, mirroring the model's overall accuracy and underscoring its robust performance across the board. Figure 15 and figure 16

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

Figure 15: Classification report of the Random Forest model

```
Random Forest - Train Data F1 score: 0.9955301269443948
Random Forest - Test Data F1 score: 0.9121238542365303
```

Figure 16: Random Forest F1 scores for Training and Test Data

C. XGBoost

The XGBoost model demonstrates precision, recall, and F1-scores that are closely aligned with those of the Random Forest model across the primary classes, however with a marginal decline in recall for class 130. This model exhibits an accuracy of 86.5%, denoting correct predictions, which positions it slightly below the CNN and Random Forest models in terms of overall performance. Additionally, the weighted average for both the F1-score and recall for the XGBoost model stands at approximately 0.88, reflecting a slight reduction in comparison to its counterparts. This indicates a nuanced yet noteworthy differentiation in the model's ability to generalize across various classes, particularly in its handling of class 130.

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

Figure 17: Classification report of the XGBoost model

```
XG Boost - Train Data f1 score: 0.9265894194530236
XG Boost - Test Data f1 score: 0.8861899786687562
```

Figure 17: XGBoost f1 scores for Training and Test Data

V. CONCLUSION AND FUTURE SCOPE

The experiment conducted in the project work gives a commendable insight about the features of ultrasonic sensor with regards to data collection for different objects at different distances and angles. It highlights the significance of the used sensor in order to take the raw data, which later gets transformed into FFT data by the programmable software. At a later stage of the project, different ML models have been developed to process the extracted information from the sensor, with different accuracy and F1 score.

The comprehensive evaluation of the CNN, Random Forest, and XGBoost models, particularly focusing on the dataset for objects at a distance of 1 meter, has unveiled insightful distinctions in the capabilities and limitations inherent to each modeling technique. Notably, the CNN model distinguished itself as the pre-eminent choice, primarily attributed to its adeptness at capturing spatial relationships within the data—an asset of immense value in the realm of signal processing tasks. Meanwhile, the Random Forest model showcased notable accuracy, underscoring the efficacy and resilience of ensemble strategies in tackling such analytical challenges. Although the XGBoost model slightly lagged in performance, it still presented a commendable accuracy level, asserting its viability as a formidable contender, especially when factors like computational efficiency and scalability are considered.

A. Acceleration of Decision Speed

Research Exploration: An in-depth analysis of optimization techniques, aimed at expediting the decision-making process within ultrasonic measurement systems, is conducted in this project work. The focus is always on identifying

methodologies that can reduce processing times without sacrificing accuracy.

Strategic Implementation: Based on the comprehensive research conducted, two principal strategies are implemented to enhance decision speed. Parallel processing methods are employed to distribute computational tasks across multiple cores or processors, enabling simultaneous execution and thereby reducing overall processing time. Additionally, algorithms within the system are optimized to minimize computational demands and improve efficiency.

Performance Evaluation: The effectiveness of the optimization efforts is assessed through real-time performance evaluations of the enhanced system. It is found that a significant reduction in processing times is achieved compared to prior versions of the system, resulting in notably faster decision-making capabilities.

Utilization of Kaggle's Computational Resources: The decision to leverage Kaggle's computational platforms for model training is made in response to the extended durations required for training on local machines. This approach allows for the reduction of training time to approximately 12 hours, highlighting the importance of external computational resources in enhancing system performance.

B. Augmentation of Measurement Accuracy

Accuracy Enhancement Research: Investigations are undertaken to discover and evaluate methods that can improve the measurement precision of ultrasonic systems. This involves reviewing a wide range of techniques from academic literature and industry practices, aimed at addressing challenges such as noise, interference, and signal distortion.

Implementation of Accuracy Improvements: A comprehensive strategy is deployed to enhance measurement accuracy, incorporating advanced signal processing techniques and noise reduction algorithms. Among these are adaptive filtering, wavelet transformation, and frequency domain analysis, which are integrated into the system based on findings from the research.

Integration of Machine Learning for Enhanced Precision: The application of machine learning technologies, specifically Convolutional Neural Networks (CNN), Random Forest, and XGBoost models, is explored. These models are trained on carefully selected datasets, aiming to utilize their pattern recognition capabilities to increase the accuracy of echo detection.

Comparative Analysis and Results: The impact of the accuracy enhancement initiatives is determined through comparative analyses using real-world data. These analyses compare the performance of the improved system against previous iterations and conventional methods. The results demonstrate a significant improvement in echo detection precision, leading to an overall increase in system accuracy.

C. Future Scope of enhancement

To further enhance the robustness and applicability of the models, a multi-faceted approach to future development is proposed. An expansion of the dataset to include a wider variety of scenarios is crucial. Introducing signals reflected from hard objects at extended distances, such as 50 meters, along with data from soft objects in varied postures, such as sitting and standing, would significantly enrich the dataset. This augmented diversity is anticipated to bolster the models' generalization capabilities and improve their accuracy by providing a more comprehensive range of data for training.

In the realm of model refinement, additional optimization of model hyperparameters presents a promising avenue for enhancing performance. The adoption of sophisticated optimization techniques, such as Bayesian Optimization, alongside the exploration of more complex CNN architectures and advanced ensemble methods, holds the potential to uncover further improvements. Moreover, the development of intricate features that more accurately capture the nuances of signal reflections from diverse surfaces and distances could significantly elevate the models' discriminative power.

Addressing the issue of class imbalance through strategies such as synthetic minority oversampling (SMOTE) or adaptive sampling is another critical area of focus. This approach is expected to enhance model performance, particularly for underrepresented classes. Concurrently, adapting the models for real-time signal processing applications, such as autonomous driving or active surveillance, could substantially increase their utility. The creation of model ensembles, which combine predictions from multiple models, could also capitalize on the strengths of each individual model to achieve superior overall performance.

Lastly, investing in frameworks that improve the interpretability and explainability of the models will be essential, especially in applications requiring justified decision-making. The application of transfer learning, especially for CNN models, by fine-tuning models pre-trained on extensive datasets to the specific dataset used in this study, could yield significant benefits. Additionally, optimizing models for computational efficiency to facilitate deployment on specialized hardware, along with extended validation through testing on an external validation set, will provide a more robust measure of predictive performance. Pursuing these strategic enhancements will not only refine the models' predictive accuracy but also extend their practical application and reliability in real-world settings, marking a critical step in the ongoing evolution of AI systems toward achieving high performance and trustworthiness.

VI. ACKNOWLEDGMENT

We would like to sincerely express our gratitude to Prof. Dr. Andreas Pech for his invaluable time in providing us lectures on “Machine learning” and helping us throughout with the various algorithms, formulae and ML models used in the project work. We also wish to extend our appreciation to Prof. Julian Umansky for his significant contributions in providing all the essential tools for the task's successful completion. Last but not the least, we are grateful to Prof. Dr. Peter Nauth for giving us the lectures on “Autonomous Intelligent System”, combined with his guidance and feedback, that ultimately provided us the right direction to finish the required project.

VII. REFERENCES

[1] Ivan Koudar, Echo detection, https://patents.google.com/patent/EP2255446A1/en, 2019

[2] Liao Qiang, Design and Implementation of Digital Ultrasonic Flaw Detector, vol. 4, 2009.

[3] 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.

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

[5] W.T. Cochran; J.W. Cooley; D.L. Favin; H.D. Helms; R.A. Kaenel; W.W. Lang; G.C. Maling; D.E. Nelson; C.M. Rader; P.D Welch, https://ieeexplore.ieee.org/document/1447887/authors, 1967

[6] W. Gentleman, G. Sande, Fast Fourier Transforms: for fun and profit, published in AFIPS '66 (Fall), November 1966

[7] Aniruddha Bhandari, Understanding & Interpreting Confusion Matrix in Machine Learning, https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/, 2024

[8] Venkatesan, Ragav; Li, Baoxin, Convolutional Neural Networks in Visual Computing: A Concise Guide, 2017

[9] Balas, Valentina E.; Kumar, Raghvendra; Srivastava, Rajshree, Recent Trends and Advances in Artificial Intelligence and Internet of Things, 2019

[10] Milecia McGregor, What Is a Convolutional Neural Network? A Beginner's Tutorial for Machine Learning and Deep Learning, https://www.freecodecamp.org/news/convolutional-neural-network-tutorial-for-beginners/, 2021

[11] Sruthi E R, Understand Random Forest Algorithms With Examples, https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/, 2024