MILESTONE 1

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,
Indranil Saha 1427190 indranil.saha@stud.fra-uas.de

***Abstract*—This study focuses on the reliability testing and enhancement of a FIUS sensor system designed for different object detection. An evaluation and improvement goal are set for the current measuring system, which is essential for detecting echo in a pulsed ultrasonic beam. The study addresses challenges related to reliability, error propagation, and modeling signal behavior. Our project uses sophisticated mathematical models and machine learning techniques to improve ultrasonic signal modelling using a FIUS system with a transducer and Red Pitaya microcontroller. Through several dependability testing, the research attempts to pinpoint shortcomings and difficulties in the existing system and contributes to the optimization of sensor systems where object detection is concerned.**

Keywords—FIUS sensor, Red Pitaya, Machine Learning, Object detection

Introduction

In the realm of autonomous vehicles navigating urban landscapes, there is a significant difficulty in identifying objects from people, especially when it comes to identifying soft from hard objects. Making this vital distinction is important for ensuring that autonomous vehicles can navigate safely and successfully requires The difficulty of this task arises from the need to accurately recognise and react to a wide range of items, each having unique physical characteristic. A key element in ultrasonic applications, echo detection plays a major role in tackling this problem and has broad ramifications in a variety of fields. The fundamental ideas behind ultrasonic echo detection, which are crucial for avoiding obstacles and measuring distance, have influenced developments in a number of sectors, including industrial automation, robotics, and navigation systems. The purpose of this research is to better understand the complexities of echo detection in order to aid in the advancement of technologies that improve the effectiveness and safety of autonomous cars in urban settings.

The main goals of our suggested system is construction of a reliable test for echo detection and the enhancement of the sensor system in conjunction with machine learning techniques.

Ease of Use

Measuring Equipments

The main tools utilised to carry out this study will be covered in this section.

The measurement apparatus that makes up our FIUS sensor is a close integration of three primary parts:

1. Ultrasonic Sensor SRF02 with a mean frequency of 40 kHz and a power output of 150 mW (manufacturer's data) for sensing.

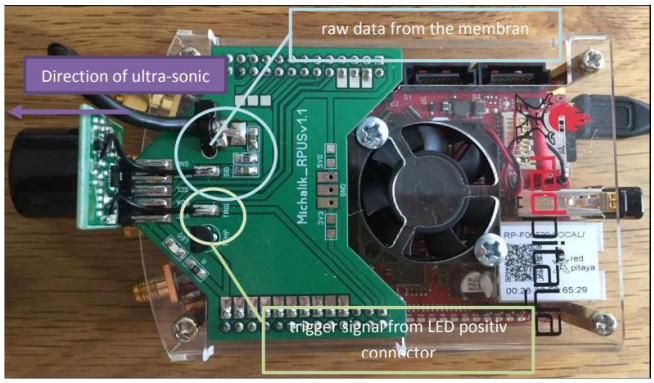


Fig. 1. Hardware of FIUS sensor

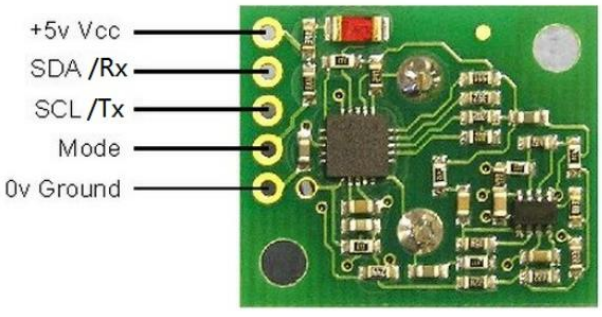


Fig. 2: Pin function of SRF02 ultrasonic sensor

2. Embedded System Red Pitaya with a sampling frequency of 1.95 MS/s and a resolution of 14 bit for analog to digital conversation of received signal.

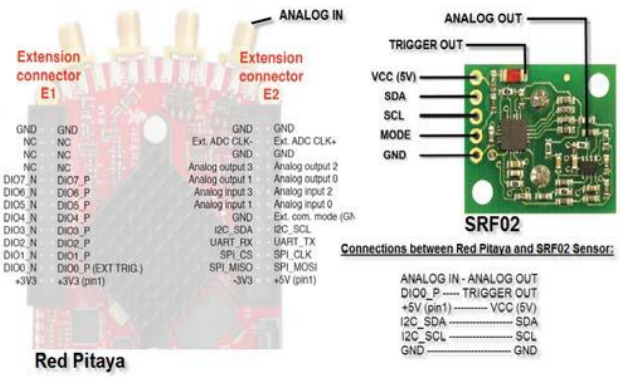


Fig. 3: Interface between Embedded System and Sensor

3. Laptop with UDP_Client Application

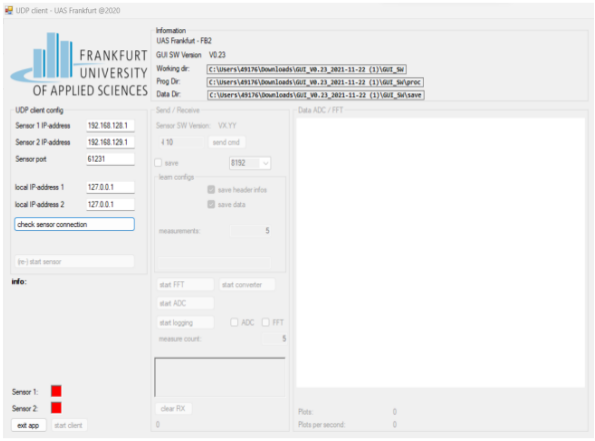


Fig. 4: UDP client view on laptop

Measuring Methods

The study has been executed using the experimental setup provided in the Robolab of Frankfurt University of applied sciences.

Over a long, black metal platform, the FIUS sensor is fixed to the top. The measurement object is set up on a white, short stand directly in front of the FIUS sensor.

1. Detection of Hard object:

To do this, we used a measuring scale to place a hard box beneath the sensor at a distance of 1 metre and 15 centimetres. We collected 1000 ADC data five times, noting the difference between the manually measured distances and those displayed on the UDP_client application. The manufacturer's data indicates that the Ultrasonic Sensor SRF02 has a mean frequency of 40 kHz and a power output of 150 mW for sensing.

Fig. 5: Snap of Hard object setup

2. Detection of Soft object (Human):

Using a measuring scale, a human stood beneath the sensor at a distance of one metre and fifteen centimetres. We collected 1000 ADC data five times, noting the difference between the manually measured distances and those displayed on the UDP_client application.

Fig. 6: Snap of Hard object setup

Literature review on ultrasonic distance measurement

With its rich history and ongoing progress, ultrasonic distance measurement is a dynamic area which has undergone significant growth over the years, with current advances in signal processing and integration with other sensor

technologies making it a vital tool in a variety of applications, promising further advancements in accuracy, reliability, and versatility.

Some of distance measurement methods using ultrasonic sensors are discussed below:

1.Time-of-Flight principle:
The Time-of-Flight principle (ToF) is a method for measuring the distance between a sensor and an object, based on the time difference between the emission of a signal and its return to the sensor, after being reflected by an object. Various types of signals (also called carriers) can be used with the Time-of-Flight principle, the most common being sound and light [1].

2.Ultrasonic Pulse-Echo methods:
The *pulse-echo method* uses an ultrasonic pulsed. Ultrasonic waves are mechanical vibrations and have a frequency greater than 20 000 Hz. Depending of the velocity of sound in a material and the frequency, the wavelength can be calculated:

$$\lambda = c/f$$

λ: wavelength [m], c: velocity of sound [m/s] and f: frequency [Hz].

An ultrasonic pulsed wave is a synthesis of sinusoidal waves with different frequencies and amplitudes [2].

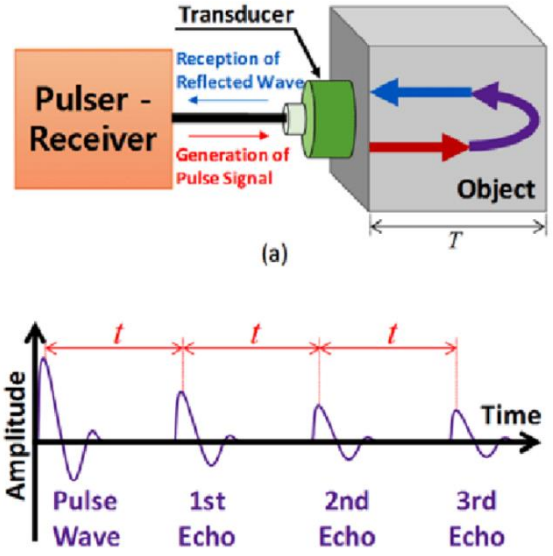


FIG.7: (A) PULSE-ECHO METHOD AND (B) A RECEIVED PULSE-ECHO SIGNAL FOR THE OBJECT [3]

3. Frequency Modulation Continuous Wave (FMCW):

It is a special type of radar system that measures both distance and velocity of moving objects. This is achieved by continuously varying the frequency of the transmitted signal by a modulating signal at a known rate over a fixed time period. A variety of frequency modulation techniques, such as sawtooth modulation, triangular modulation, sine wave modulation can be used to change the frequency pattern of the emitted radio wave [4].

4.Triangulation Methods:

It involves triangulating the distance to an object by using several ultrasonic sensors positioned at specified angles and distances from the target. It is Frequently utilised in positioning systems and mobile robotics.

ML method used for finding first reflection

In machine learning (ML), finding the initial reflection usually entails using algorithms and approaches that can analyse and interpret data to find pertinent patterns or features associated with reflections.

An overview of how machine learning could be used to locate the first reflection is provided below:

1.Data Collection:
Data collection is a crucial step in the machine learning (ML) pipeline, as the quality and quantity of the data directly impact the performance of the trained model. Few steps involves in data collection processes are identify data sources, data requirements, ethical and legal consideration, data sampling etc.

2.Feature Extraction:
Feature extraction is a process in machine learning and data analysis that involves identifying and extracting relevant features from raw data. These features are later used to create a more informative dataset, which can be further utilized for various tasks such as Classification, Prediction, Clustering. Feature extraction aims to reduce data complexity while retaining as much relevant information as possible. This helps to improve the performance and efficiency of machine learning algorithms and simplify the analysis process [5].

3.Training Data Preparation:
Label a subset of the collected data to create a training dataset. For each data point, indicate whether it corresponds to the presence or absence of the first reflection.

4.ML Algorithm Selection:
Finding patterns in data and using that information to make precise predictions is the aim of machine learning, an algorithm-based data analysis technique. ML algorithms, as their name implies, are essentially computers that have been educated in various ways. There are three major types of ML algorithms: unsupervised, supervised, and reinforcement.

5.Training the Model:
Train the selected ML model using the labeled training dataset. The model learns to associate the identified features with the presence or absence of the first reflection.

6.Testing the Model:
Determine the initial reflection by applying the trained model to fresh, unobserved data. Predictions based on the patterns that the model has learned will be produced.

7.Evaluation:
Evaluate the model's performance by contrasting the labels that it predicts with the ground truth. Evaluation criteria that are frequently used include F1 score, recall, accuracy, and precision.

REFERENCES

[1] TERABEE, Blog on Time-of-Flight principle

[2] Laura-Kristin Scholtz, Ultrasonic Pulse-Echo Method, January 2016.

[3] Hogeon Seo, Dong-Gi Song, Kyung-Young Jhang, "Journal of the Korean Society for Nondestructive Testing", April 2016

[4] Editorial Team by everything RF, Technical Article, May 8, 2022

[5] https://domino.ai/data-science-dictionary/feature-extraction

MILESTONE 2

Reliability test and improvement of a sensor system for object detection

Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth & Dr. Andreas Pech

Deblina Karmakar
1427365
deblina.karmakar@stud.fra-uas.de

Indranil Saha
1427190
indranil.saha@stud.fra-uas.de

Data Setup:

1. Data Collection

1.1 Hard Surface Measurements

We measured for hard surfaces at two distances: 1 meter (m) and 50 centimetres (cm). The measurements were taken to assess. For each distance, we obtained 1000 measurements.

Distance:

1 meter (m)

50 centimetres (cm)

Number of Measurements:

1000 readings for each distance

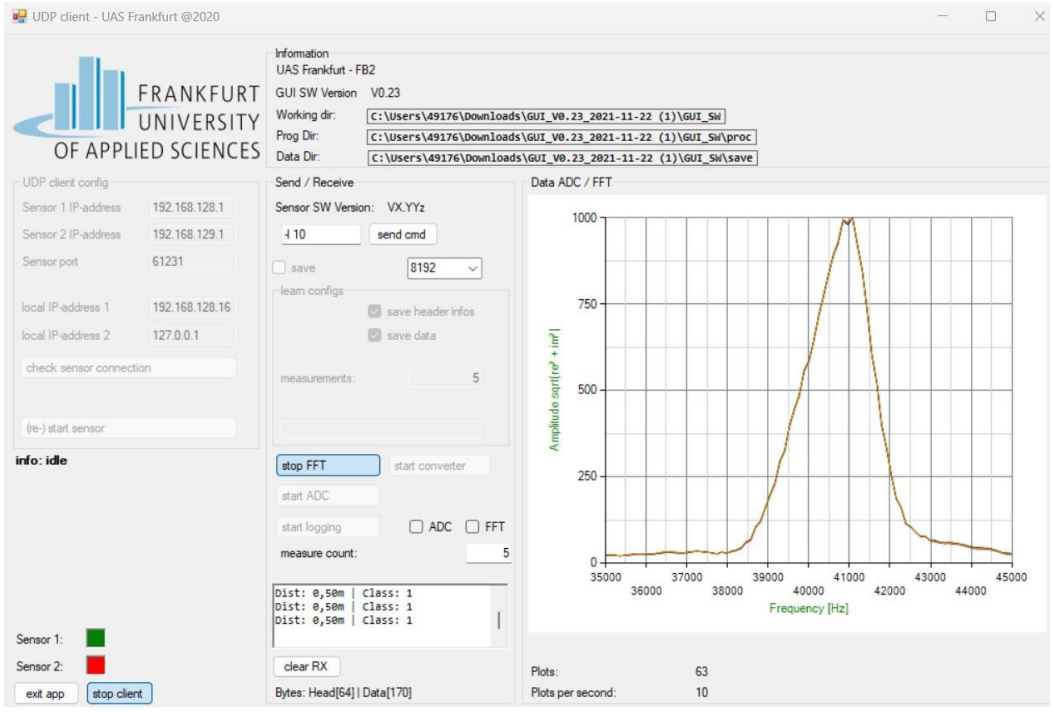
1.2 Soft Surface (Person) Measurements

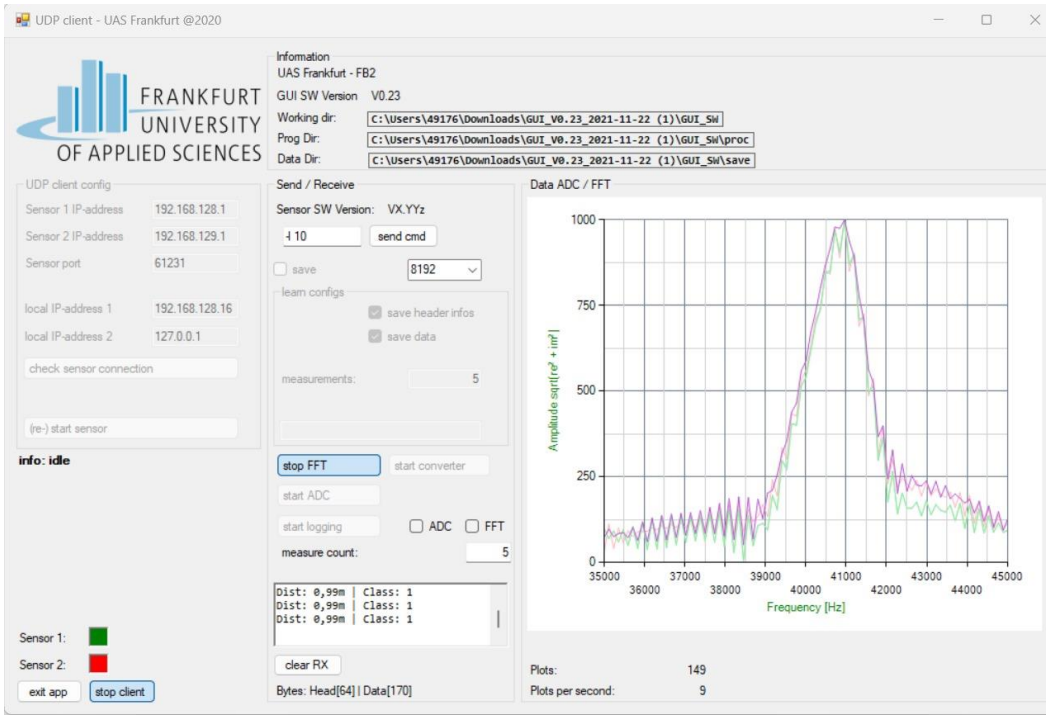
Similar measurements were taken on soft surfaces to take the readings. The following details outline the data collection process:

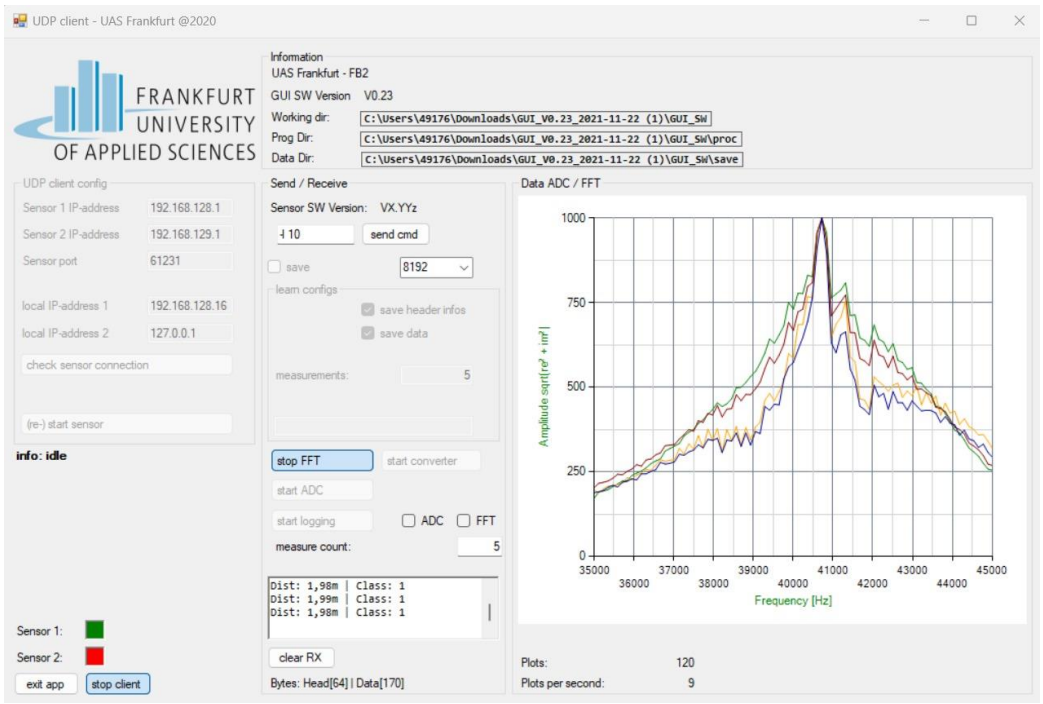
Number of Measurements:

1000 readings

1. **FIUS inbuilt distance measurement (dFIUS):**







2. Manually measured distance using a folding meter stick (dMAN):

Comparison:

dFIUS	**dMAN**
1m	0.99m
50cm	0.50m

Python program that automatically calculates and displays the distance dML between the sensor and the first reflection using the ML first echo detection algorithm given using the Source code.

Approach I:

```
import smbus

import time

import numpy as np

from scipy.signal import find_peaks


# Constants

I2C_ADDRESS = 0x70

V_SONIC_WAVE = 343.2  # Speed of sound in m/s

ADC_START_DELAY_US = int(0.30 * 2 * 1e6 / V_SONIC_WAVE)

ADC_BUFFER_DELAY_US = int((16_384 * 512) / 1e3)  # Assuming ADC_BUFFER_SIZE is 16384

ADC_MID_US = ADC_START_DELAY_US + (ADC_BUFFER_DELAY_US // 2)


# Setup I2C

bus = smbus.SMBus(1)  # Typically 1 for /dev/i2c-1
```

```
def start_ultrasonic():
    # Write to I2C to start ultrasonic measurement
    bus.write_byte_data(I2C_ADDRESS, 0, 0x52)  # 0x52 is a placeholder command

def read_ultrasonic():
    # Read from I2C
    high_byte = bus.read_byte_data(I2C_ADDRESS, 1)
    low_byte = bus.read_byte_data(I2C_ADDRESS, 2)
    time_of_flight_us = (high_byte << 8) | low_byte
    return time_of_flight_us

def get_adc_data():
    # Placeholder for actual ADC data acquisition
    # Replace with actual ADC reading code
    return np.random.randn(16_384)  # Random data to simulate ADC data

def calculate_distance(time_of_flight_us):
    return V_SONIC_WAVE * (time_of_flight_us / 2e6)

def measure_distance():
    start_ultrasonic()
    time.sleep(ADC_START_DELAY_US / 1e6)  # Convert microseconds to seconds
    adc_data = get_adc_data()
    time.sleep(ADC_BUFFER_DELAY_US / 1e6)  # Wait for ADC buffer to fill
    time_of_flight_us = read_ultrasonic()
    distance = calculate_distance(time_of_flight_us)
    return distance

# Example usage
if __name__ == '__main__':
    distance = measure_distance()
    print(f"Measured distance: {distance} meters")
```

Approach II:

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from redpitaya.overlay.mercury import mercury as overlay

def ml_first_echo_detection(signal, sampling_rate):
    # ML first echo detection algorithm
    threshold = 0.5  # Adjust threshold based on your specific use case
    peaks, _ = find_peaks(signal, height=threshold)

    if len(peaks) > 0:
        return peaks[0] / sampling_rate
    else:
        return None

def main():
    # Initialize Red Pitaya overlay
    ol = overlay()
    rp = ol.red_pitaya

    # Parameters
    sampling_rate = int(rp.fpga.api.osc_frequencies_get(0))

    # Configure and acquire data from channel 1 (assuming ultrasonic sensor is connected to channel 1)
    rp.reset()
    rp.waveform_acquisition_start()
    rp.waveform_acquisition_trigger()

    # Wait for acquisition to complete
    while rp.waveform_acquisition_completed() == 0:
        pass

    # Retrieve acquired data
    data = rp.waveform_acquisition_data(0, -1)

    # Apply ML first echo detection algorithm
    detected_distance = ml_first_echo_detection(data, sampling_rate)
```

```
    # Display results
    if detected_distance is not None:
        print(f"ML First Echo Detected at a distance of {detected_distance:.4f} meters.")
    else:
        print("No echo detected.")

    # Plot the acquired data
    plt.plot(data)
    plt.title('Acquired Ultrasonic Signal')
    plt.xlabel('Time (samples)')
    plt.ylabel('Amplitude')
    plt.show()

if _name_ == "_main_":
    main()
```

MILESTONE 3

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

Deblina Karmakar
Matriculation: 1427365
deblina.karmakar@stud.fra-uas.de

Indranil Saha
Matriculation:1427190
indranil.saha@stud.fra-uas.de

Abstract: The purpose of this study was to evaluate the precision of a custom-developed software for measuring distances, using manual measurements as a benchmark. For this analysis, Data set #2 was employed. The procedure involved comparing manually obtained distances with those generated by the software. To assess the software's accuracy, a confusion matrix was utilized, categorizing discrepancies less than 1 cm as successful matches and those greater than 1 cm as discrepancies. The findings demonstrated the software's commendable precision, with most measurements lying within the acceptable margin of error. The confusion matrix offered a detailed insight into the software's capability to measure distances accurately, underscoring its potential utility and dependability for various applications.

I. INTRODUCTION

In numerous sectors, such as industrial processes, robotics, and navigational systems, the precision of distance measurements is paramount. The evolution of technological capabilities has paved the way for software-driven methods of determining distances, presenting a more convenient and efficient alternative to conventional manual techniques. Nevertheless, the dependability and precision of these software solutions must undergo thorough verification to confirm their fitness for real-world application.

This research was conducted to scrutinize the efficacy of a bespoke software tool designed for distance measurement, focusing on data set #2 for this purpose. The study aimed to juxtapose the distances calculated by this software against those obtained through manual measurement to ascertain the software's precision. A significant aspect of the evaluation was the comparison of manual measurement distances (dMAN) with those computed by the software (dML, dFIUS), with deviations under 1 cm being considered satisfactory and those beyond this limit signaling potential measurement errors.

The intent of this investigation was to shed light on the software's utility and dependability for measuring distances. The outcomes of this study enrich the collective knowledge on the viability of software-based tools for distance measurement in practical settings. Moreover, the methodology applied in this research, including the development of a confusion matrix, lays down a comprehensive framework for the future examination of software intended for distance calculation.

II. METHODOLOGY

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.

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.

Results:

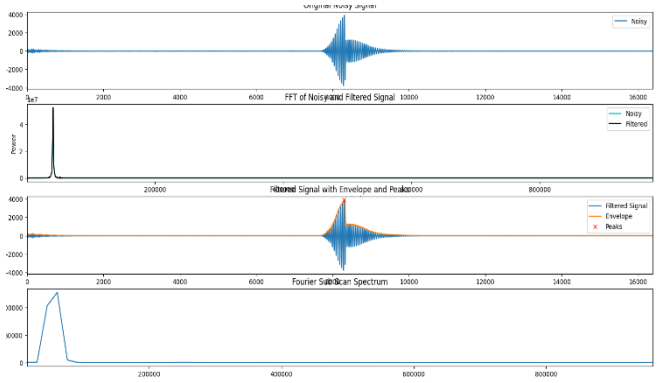


Figure 1: ADC to FFT plot for object placed at 1m

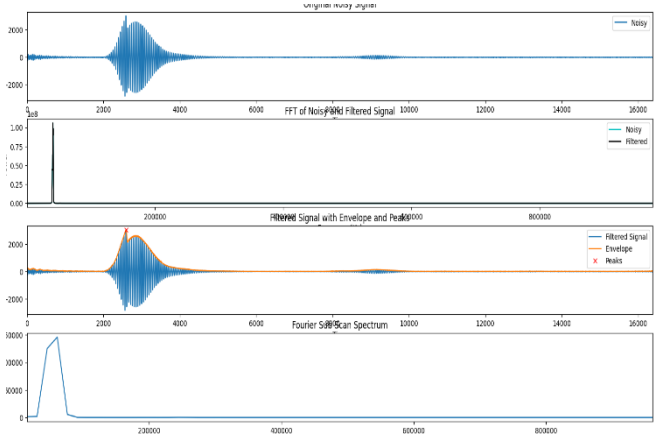


Figure 2: ADC to FFT plot for an object placed at 50cm

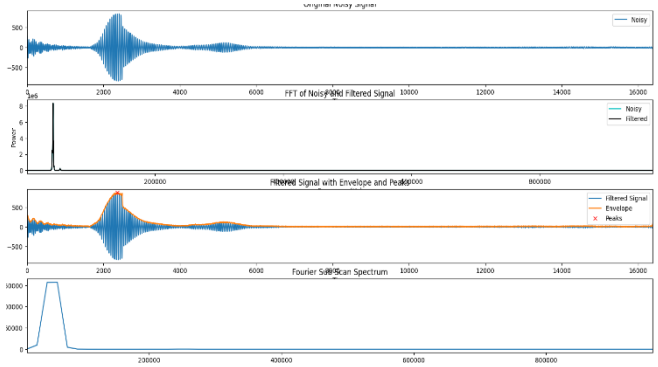


Figure 3: ADC to FFT plot for soft object standing

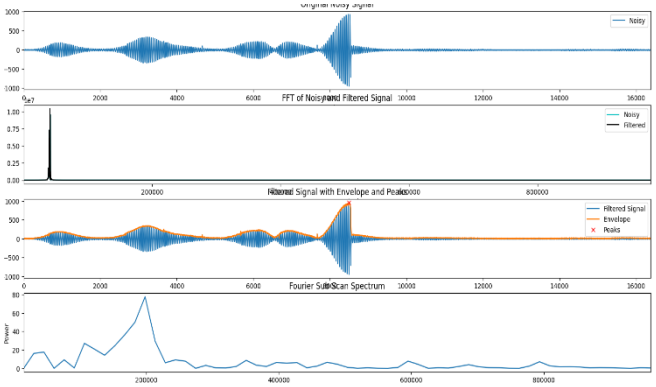


Figure 4: ADC to FFT plot for soft object sitting

III. 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

MILESTONE 4

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

Deblina Karmakar
Matriculation: 1427365
deblina.karmakar@stud.fra-uas.de

Indranil Saha
Matriculation: 1427190
indranil.saha@stud.fra-uas.de

Abstract: This subsequent research outlines an elaborate strategy for improving the accuracy of echo detection in ultrasonic sensors. By integrating techniques such as Convolutional Neural Networks (CNN), Random Forests, and XGBoost, this study proposes a diverse machine learning approach to better identify the initial echo signal. Utilizing the Red Pitaya board along with the Ultrasonic Sensor SRF02, our attention is centered on dataset #3 to test our proposed methods. This document elaborates on the software solutions deployed and examines the effectiveness of each algorithm, signifying a notable progress in the field of echo detection technology.

Keywords—Echo Detection, Ultrasonic Sensing, Machine Learning, Convolutional Neural Networks, Random Forest, XGBoost, Precision Enhancement

I. INTRODUCTION

In this study, we delve deeper into the use of advanced machine learning techniques to improve the precision of detecting the initial echo in ultrasonic sensor systems, building on the foundation laid by our earlier work. We adopt a comprehensive approach that integrates a variety of machine learning models, each chosen for their unique strengths and synergistic potential. Convolutional Neural Networks (CNN) are utilized for their superior ability in processing signals, crucial for identifying patterns in complex data sets. Random Forests are selected for their robustness in classification, ensuring reliable performance across different scenarios. Furthermore, XGBoost is employed for its efficiency and effectiveness in predictive modeling, offering a rapid and powerful solution for analysis.

Our objective is to set new benchmarks in the accuracy of echo detection, which is vital for various applications, from autonomous vehicles to sophisticated safety systems. Our methodology focuses on improving the precision of detecting the first echo by meticulously identifying the highest peak within specific time frames. These frames are intelligently defined by our machine learning models, allowing for the precise localization of the echo's most significant aspect with unprecedented accuracy.

To test and validate our advanced detection techniques, we turn our attention to dataset #3, which will act as a critical testbed for our refined algorithms. Here, we expect to see marked enhancements in detection performance. Through continuous experimentation and improvement, our research aims to significantly advance the field, exploring new frontiers in ultrasonic sensor technology and echo detection capabilities.

II. METHODOLOGY

The approach undertaken leverages a two-tiered strategy that initially employs machine learning to determine specific time windows, followed by a focused analysis to identify peak signals. This section delves into the implementation details of the CNN, Random Forest, and XGBoost models, shedding light on their training processes, evaluation methods, and their deployment in pinpointing echo signals.

A. Convolutional Neural Networks (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.

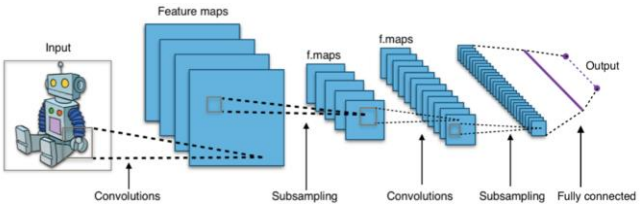


Fig.1. A typical CNN Architecture

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.

B. *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.

C. *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.

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

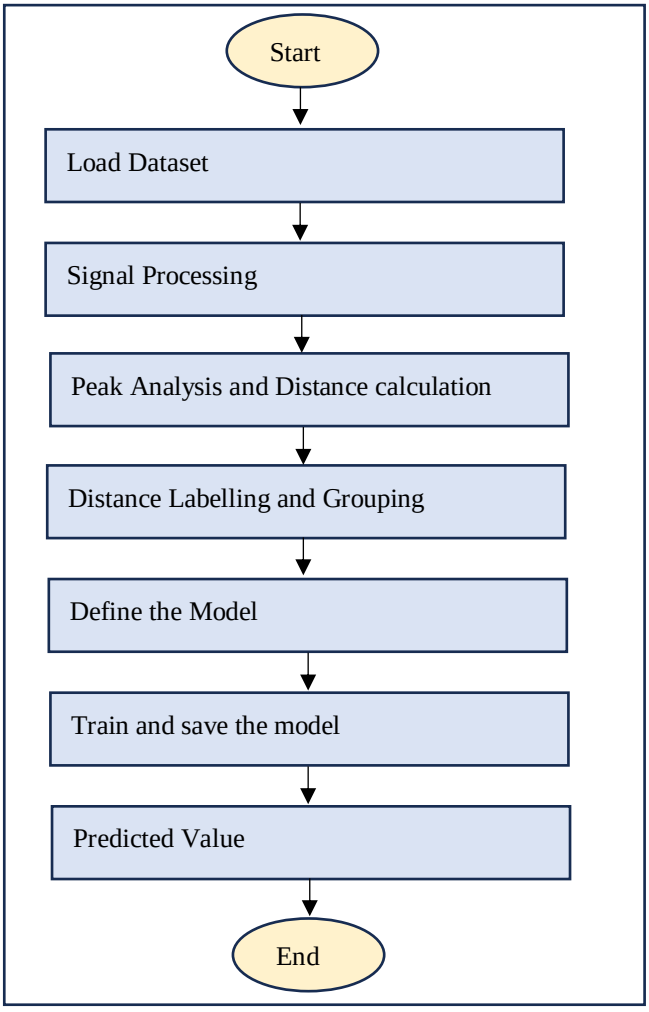
This can be done for 'm' iterations, until residuals have been minimized as much as possible.

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 the below 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.



B. Pre-Processing

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.

Training the Model

A. Convolutional Neural Networks (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.

B. 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.

C. 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 ANALYSIS:

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

A. 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.5. Classification report of CNN Model

B. 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.6. Classification report of Random Forest Model

C. XGBoost

The 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.

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.7. Classification report of XGBoost Model

V. CONCLUSION & FUTURE SCOPE

In conclusion, 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.

Future Scope

Moving forward, to enhance the robustness and applicability of the models, the following future directions could be considered:

Dataset Expansion: Introducing a broader range of scenarios, including signals reflected from hard objects at greater distances such as 50m, and incorporating data from soft objects with different postures (sitting and standing), would enrich the dataset. This diversity could help improve the model's generalization capabilities and accuracy.

Model Optimization: Further tuning of the model hyperparameters, perhaps through more sophisticated methods like Bayesian Optimization, could lead to better performance. Additionally, exploring more complex CNN architectures or advanced ensemble techniques could uncover improvements.

Feature Engineering: Developing more intricate features that capture the essence of signal reflections from various surfaces and distances might enhance the models' ability to distinguish between different classes more effectively.

Class Imbalance Mitigation: Addressing the class imbalance with techniques such as synthetic minority oversampling (SMOTE) or adaptive sampling could improve the model performance, particularly for underrepresented classes.

Real-time Processing: Adapting the models for real-time signal processing could be beneficial for applications that require immediate decisions, such as autonomous driving or active surveillance systems.

Model Ensemble: Combining the predictions from multiple models in an ensemble method could yield better performance than any single model, leveraging the strengths of each.

VI. 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

MILESTONE 5

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

Deblina Karmakar
Matriculation: 1427365
deblina.karmakar@stud.fra-uas.de

Indranil Saha
Matriculation: 1427190
indranil.saha@stud.fra-uas.de

Abstract: This document details improvements applied to an ultrasonic measurement system, emphasizing advancements in decision-making speed, measurement precision, and user interface. By refining algorithms, employing advanced signal processing techniques, conducting thorough analysis, and developing a user-friendly graphical interface, notable enhancements have been realized. These advancements facilitate quicker decision-making in real-time, superior accuracy in measurements, and an enhanced user experience, altogether augmenting the system's efficiency and ease of use.

Keywords—CNN, Kaggle, XGBoost

I. INTRODUCTION

Ultrasonic measurement systems play a critical role across numerous sectors, providing essential non-destructive testing functions for identifying defects and measuring distances. Enhancing these systems for greater speed and precision poses a significant challenge. This document introduces a range of breakthroughs designed to boost the ultrasonic measurement system's capabilities.

To address the demands for quicker decision-making and heightened precision, our strategy incorporated the use of sophisticated machine learning techniques, including Convolutional Neural Networks (CNN), Random Forest, and XGBoost. The computational intensity of training and testing these algorithms led us to utilize Kaggle's powerful platform, facilitating a more efficient development process.

For improved user engagement and functionality, a graphical user interface (GUI) was created using Flask, a Python-based web framework. While the GUI is operated locally for ease of use, employing Kaggle for the heavy lifting in model training significantly cut down on development time, enhancing the system's decision-making speed.

Our enhancements, which combine cutting-edge algorithms with a methodical approach and an intuitive interface, are aimed at advancing the ultrasonic measurement system's performance. By tapping into Kaggle for computational tasks and keeping user interaction local via Flask, we achieve a balance between operational efficiency and user-friendly access, setting a new standard for precision and efficiency in industrial settings.

II. METHODOLOGY

A. Decision Speed Improvement:

Research is conducted on various optimization techniques aimed at accelerating the decision speed in ultrasonic measurement systems. Methodologies are explored that could reduce processing time without compromising accuracy.

Implementation involves the application of two primary strategies to enhance decision speed. Parallel processing techniques are utilized to distribute computational tasks across multiple cores or processors, enabling concurrent execution and reducing overall processing time. Additionally, algorithms used in the system are optimized to minimize computational overhead and improve efficiency.

Data and results are measured by conducting real-time measurements with the optimized system. A significant reduction in processing time is observed compared to previous iterations. The decision-making process is noted to be more responsive, enabling quicker analysis and action.

The computational resources available on the Kaggle platform are utilized for training models. This decision is made due to the extensive time required for training on local machines. Training time, which might have taken hours locally, is reduced to approximately 12 hours on Kaggle, highlighting the substantial impact of utilizing external resources on system performance.

B. Measurement Accuracy Enhancement:

Research is focused on identifying and evaluating methods to enhance measurement accuracy in ultrasonic systems. Techniques are investigated from scientific literature and industry practices to address challenges related to noise, interference, and signal distortion.

Implementation involves a comprehensive approach to improve measurement accuracy. Advanced signal processing techniques and noise reduction algorithms are integrated into the system. These include adaptive filtering, wavelet transforms, and frequency domain analysis, among others.

Machine learning algorithms are applied to further enhance accuracy. Convolutional Neural Networks (CNN), Random Forest, and XGBoost are utilized to analyze and classify ultrasonic signals. These models are trained on labeled

datasets, aiming to leverage their pattern recognition capabilities for improving echo detection accuracy.

Effectiveness is assessed through comparative measurements using real-world data. The performance of the enhanced system is compared against previous iterations and benchmarked against traditional methods. A notable improvement in the precision of echo detection is observed, leading to enhanced overall accuracy.

Screenshots:

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 1: 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 2: CNN Accuracy

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

Figure 3:XGBoost Classification Report

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

Figure 4:XGBoost Train Test F1 Score

Random Forest:

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 5:Random Forest Classification Report

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

Figure 6:Random Forest Train Test F1 Score

C. Pending Implementation:

1. Software for Cropping the First Echo's Data:

Although the concept for software that isolates the data of the initial echo has been designed, its full development and deployment remain incomplete. The purpose of this software is to identify the first echo's location within an ultrasonic signal and to segment a specific portion of the time signal around the echo for additional examination. It is intended to feature customizable settings, allowing users to define the duration and positioning of the time window to be preserved.

2. Thorough Examination of Outcomes, Data, and Fast Fourier Transforms:

A thorough examination involving outcomes, data, and Fast Fourier Transforms (FFTs) is yet to be performed. This examination is designed to assess the ultrasonic measurement system's efficacy, particularly concerning incorrect predictions. Through a detailed review of the outcomes and FFTs, insights into the causes of incorrect predictions are anticipated. These insights will be drawn from an extensive review of ultrasonic physics literature, including scientific journals and books.

III. 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