Reliability Test and Improvement of a Sensor System for Object Detection

Machine Learning
Under Prof. Andreas Pech

Anurag De
1400450
anurag.de@stud.fra-uas.de

Gaurav Honnavara Manjunath
1384178
gaurav.honnavaramanjunath@stud.fra-uas.de

Md Mosharraf Hossain
1386448
md.hossain3@stud.fra-uas.de

Md Zahid Hasan
1396470
md.hasan5@stud.fra-uas.de

Abstract- *This paper is a brief analysis focused on our approach to improve reliability and accuracy of red-pitaya sensor system. It gives an overview of the sensor system for object detection by employing advanced ultrasonic signal analysis. The primary goal of this application is to detect the position of the first echo i.e. the first reflection of a pulsed ultrasonic beam. The Ultrasonic Proximity sensor facilitates this by converting high-frequency sound waves into an analog to digital conversion (ADC) data signal over a designated time frame.*

A machine learning (ML) algorithm is designed to compare calculated distances with actual distances, thereby striving to improve accuracy. Additionally, we construct a confusion matrix to analyze the deviation between the distances calculated by the ML algorithm and the actual distances, providing insights into the algorithm's performance. Along with this, an ML-driven maximum peak detection approach using convolutional neural network is designed to improve the precision of peak detection and overall reliability of the sensor system for object detection.

Keywords—FUIS, Peak Detection, Convolutional Neural Network (CNN), Signal Processing.

I. Introduction

Machine Learning, a subfield of Artificial Intelligence (AI) is one of the most impactful fields in modern times. We can also call Machine Learning a self-training model which allows computers to learn and improve their performance automatically based on various datasets. Machine Learning models can make decisions according to previously learned data with human programming [1]. Various Machine Learning methods depend on data which is taken by sensors and give perfect output that autonomous systems can use to perform complex tasks where also has one complex process which is called machine perception which is based on the machine learning algorithm (there are lot of machine learning algorithms such as SVM, Peak Detection etc.) to process the data and increase the accuracy of the models [2].

Our research involves testing and measuring the distance between the Ultrasonic sensor and the objects (soft/hard) in the real world. We design machine learning concepts to create a system that measures distances based on the data provided by the ultrasonic sensor. The goal is to identify and enhance the output of a sensor that measures the distance of objects, whether they are human or any hard object.

We utilize ADC data from the Red Pitaya Ultrasonic sensor. These measurement datasets are used to generate numerous confusion matrices for our research. We use these measurement datasets to create several confusion matrices for our analysis, and then use the values obtained in these matrices to further train the model to increase the reliability and precision of the sensor's output.

II. LITERATURE REVIEW

In recent times, object detection has become one of the most crucial tasks across various fields, particularly in the autonomous vehicle sector, where driverless technology is advancing rapidly. Numerous researchers are dedicated to enhancing various sensors to achieve the highest accuracy in object detection.

In the field of object detection, Amjoud et al. [11] provide a comprehensive review titled "Object Detection Using Deep Learning, CNNs, and Vision Transformers". In their work, they explore various deep learning algorithms including AlexNet, VGGNet, ResNets, Inception-ResNet, GoogLeNet, CSPResNeXt, among others, for object detection. They conduct their experiments on datasets such as Pascal VOC and MS-COCO. The algorithms are categorized into two types: single-stage detectors and two-stage detectors. According to their findings, they report accuracies of 57.7% and 63.1% respectively on the MS-COCO dataset.

Nesti et al. [12] present a method titled "Ultra-Sonic Sensor-based Object Detection for Autonomous Vehicles". In their proposed approach, they employ ultrasonic sensors for object detection in autonomous vehicles. They developed an end-to-end framework where they initially transform input tabular ultra-sonic sensor (USS) data into voxelized 3D point clouds. From this point, they derive multi-channel bird's-eye-view images, which are then fed into a deep learning model. Specifically, they utilize ResNet-50 and leverage the COCO dataset for training. Their model achieves an accuracy of 75.82%.

Liu et al. [13] conducted research on "Real-time Object Detection using LiDAR and Camera Fusion for Autonomous Driving". In their methodology, they utilized the largest computer vision evaluation dataset for autonomous driving, which comprises HD camera photos with 7481 training samples and 7518 testing samples collected across urban, rural, and highway environments. The image size was set at 640 * 640 pixels, and the dataset included 8 classes such as tram, miscellaneous, cyclist, etc. They employed single RGB-based, single LiDAR-based, and fusion-based algorithms in their approach. At times, they observed that both RGB and LiDAR individually missed some objects during detection, whereas the fusion-based algorithm consistently detected objects accurately.

In their work on “LiDAR-based object detection”, Meyer et al. [14] developed LaserNet, a fully convolutional neural network. This network has the capability to generate multimodal distribution 3D prediction boxes for every point within a point cloud. These distributions are subsequently amalgamated to form predictions for individual objects.

III. THEORETICAL BACKGROUND

Here we are giving an overview about the theoretical elements which we used in our models to reliability test and improvement of a sensor system for object detection. Here, we will discuss hardware requirements, the programming language used, and the machine learning algorithm we used to develop our model.

1. *Red Pitaya*

Red Pitaya device is used for our project as the sensor. It's a free and open-source gadget. This means that the sensor's functionality can be coded freely and independently based on our needs, and many previously created programs can be found on the Internet's open-source networks. It gives us a flexible measuring device that can be used with a smartphone, tablet, or PC and one or more probes to replace a variety of much more expensive separate measuring equipment.

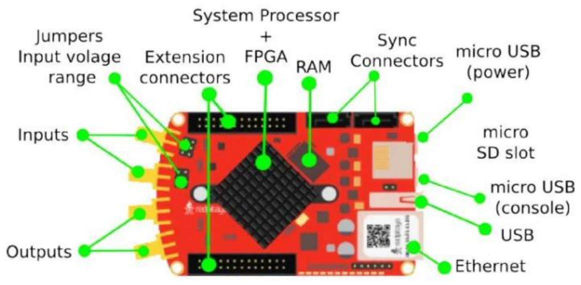


Figure 1: Red Pitaya

The Red Pitaya microchip STEM Lab board gives us high bandwidth to convert data from Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC). Because of its features, particularly as a radio receiver and transmitter, it is well liked choice for radio frequency applications. The Red Pitaya is widely used in various radio frequency applications due to its diverse range of characteristics such as Automotive, Aerospace and defense, Advance research, Quantum computing, Telecommunications, and also medical sector etc.

In our project, we used Red Pitaya sensor to carry out measurements which were given from the Lab of Frankfurt University of Applied Sciences which was preconfigured with Linux system and the UDP client program. The core Red Pitaya unit has two analog RF inputs and outputs, as well as 16 general input and output ports. The board also includes a micro-USB connector for the console, a micro-SD card slot, an RJ45 socket for Ethernet, and a USB port. The Red Pitaya can receive as well as transmit radio frequency signals. 50 MHz is the frequency range.

2. *First Fourier Transform (FFT)*

Our desired numerical discrete format is provided by the Red Pitaya server using the Fast Fourier Transform concept. Gaining a comprehensive understanding of the project deliverables requires a general understanding of the Fast Fourier transform. It extracts frequency data by separating a signal into its spectral components. FFTs are used in condition monitoring, quality control, and machine and system defect analysis.

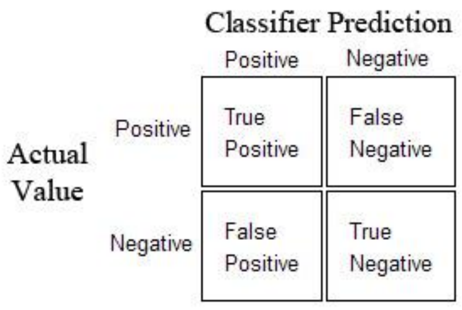
More specifically, the FFT is the best algorithm available for performing the "Discrete Fourier Transformation" (DFT). A signal is gradually sampled and divided into its frequency components. These parts consist of a single, amplitude- and phase-varying sinusoidal oscillation with a different frequency

3. Analog to Digital Converter (ADC)

ADC functionality enables the conversion of analog signals, such as those received from the sensor's ultrasonic waves, into digital data that can be processed by our machine learning algorithms. The utilization of ADC from the Red Pitaya Ultrasonic sensor ensures that we capture precise measurements.

4. Confusion Matrix

A confusion matrix offers an overview of a machine learning model's performance on a given set of test data. It serves as a tool to display the accuracy and inaccuracy of the model's predictions, particularly in the context of classification tasks where categorical labels are assigned to instances of data.



		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 2 : Confusion Matrix

This matrix plays a crucial role in evaluating the performance of classification models, providing insights into metrics such as recall, accuracy, precision, and overall effectiveness in distinguishing between classes. By detailing the counts of true positive, true negative, false positive, and false negative predictions [9], it facilitates a deeper understanding of the model's classification capabilities.

a) ***True Positive (TP):*** We have a true positive case when the expected value equals the actual value. This suggests that while the model anticipated a positive outcome, the actual outcome is also positive.
b) ***True Negative (TN):*** A situation is considered truly negative when the real value and the expected value are equal. On this occasion, though, both the actual value and the model's prediction are negative.
c) ***False Positive (FP):*** A false positive occurs when the expected value corresponds with the incorrect value. Despite the fact that the actual value was negative, the model predicted a positive outcome. This kind of prediction error is known as type one.
d) ***False Negative (FN)***: A false negative case occurs when the expected value coincides with the incorrect value. The model predicted a negative result, but the actual number was positive. This is the prediction of type two error, and it helps us find more parameters.
e) ***Accuracy:*** The model's accuracy is used to measure its performance. It is calculated as the proportion of all correct instances to all instances.
f) ***Precision:*** Precision evaluates the accuracy of a model's positive predictions. It is calculated as the ratio of true positive predictions to the sum of true positive and false positive predictions made by the model.
g) ***Recall:*** The recall of a classification model measures its ability to correctly identify all relevant instances within a dataset. It is calculated as the ratio of true positive (TP) cases to the sum of false negative (FN) and true positive (TP) cases.
h) ***F1 Score:*** The F1 score is used to assess a classification model's overall performance. It is the harmonic means of precision and recall.

$$F1\ Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

5. Peak Detection using Ultrasonic sensor - The ultrasonic sensor operates based on the principle of wave propagation. It emits ultrasonic waves into its surrounding environment, and these waves reflect off materials with different acoustic impedances compared to air. The rate at which these waves reflect varies depending on the surface characteristics. For instance, if the surface is rough relative to the wavelength, the wave not only reflects but also scatters and backscatters. The reflected and backscattered components of the wave interact, incorporating the shape and roughness of the surface onto the wave. As a result, the received signal differs significantly from the original signal emitted during the burst, providing information about the properties of the reflecting surface.

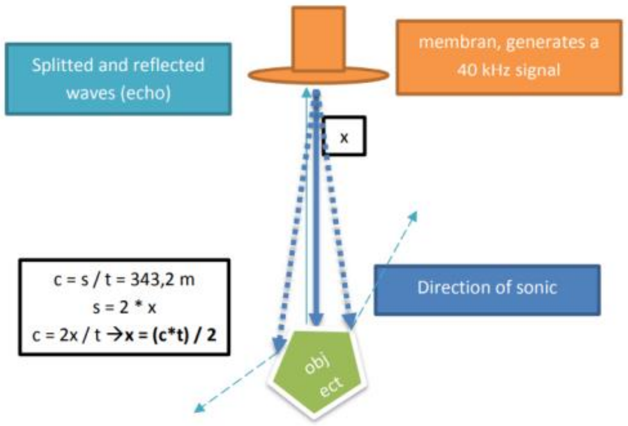

Splitted and reflected waves (echo)
membran, generates a 40 kHz signal
x
c = s / t = 343,2 m
s = 2 * x
c = 2x / t →x = (c*t) / 2
Direction of sonic
obj ect

Figure 3:Ultrasonic pulse echo method.

The sensor comprises three key components: an ultrasonic transducer, an embedded system responsible for transducer control and signal acquisition, and a dedicated computer for in-depth signal analysis, feature extraction, and object classification.

The sampled signal is to be captured by interfacing the ultrasonic transducer with an Embedded system (MUC)/ a development kit. The captured signal from the MCU needs to be used further for signal analysis. The software is designed to incorporate different state machines, including Initialization, Data Acquisition, Echo Detection, Distance Calculation, and Feature Extraction from the acquired signal. Features are extracted from both the Fourier domain and the time domain. The acquired samples are utilized to compute the mean and variance of three features: Signal-to-Noise and Distortion Ratio (SINAD), the location of the first peak in the spectrum, and the number of peaks.

Initially, ultrasonic samples are collected and SINAD (Signal-to-Noise and Distortion Ratio) is computed. Peak detection is performed by applying Fast Fourier Transformation (FFT) using a Hanning Filter. This data is then passed to the "Peak Detection" function, which determines the locations and number of peaks in the selected spectrum. Mean and variance calculations are subsequently carried out based on the number of peaks detected.

Classification can be performed based on parameters such as average peak detections. For instance, a low number of peaks and a peak frequency close to the fundamental frequency of 40 kHz may indicate that the surface of the scanned objects is smooth. Conversely, if the signal spectra exhibit average peaks and a first peak below the fundamental frequency, it suggests a different surface characteristic.

6. Distance (d_{ML}) Measurement using ML driven first echo detection algorithm - This section explains the method for calculating distances from the sensor using first peak echo detection with the provided measurement from the Analog to Digital Conversion (ADC) data and signal processing techniques, this approach enables precise distance estimation between the sensor and objects in the environment. The distance calculation relies on the time delay between the transmission and reception of ultrasonic pulses. The formula used for distance (d) calculations is:

$$d = \frac{v \, x \, t}{2}$$

d represents the distance between the sensor and the object. v denotes the speed of sound in air (typically around 343.2 m/s.), t signifies the time delay between transmission and reception of the ultrasonic pulse. The time delay (t) is inferred from the position of the first peak in the ADC data. This approach assumes a constant speed of sound in air and accounts for the round-trip time of the ultrasonic pulse.

The peak detection process enables the identification of key features in the ADC data, specifically the peaks corresponding to the received ultrasonic signals. By filtering out noise and selecting only valid peaks above the threshold, the function ensures that accurate distance measurements can be derived from the data. The first step involves identifying peaks in the ADC data, which represent significant fluctuations or high-amplitude points in the signal. This is accomplished using the ***find_peaks*** function, which detects local maxima in the data. To distinguish meaningful peaks from noise, a relative threshold is calculated based on a percentage of the mean value of the ADC data. Peaks with amplitudes above this threshold are considered valid and retained for further analysis. Peaks that do not meet the threshold criteria are filtered out, leaving only the valid peaks that correspond to the received ultrasonic signals. These valid peaks serve as reference points for subsequent distance calculations. In the ADC graph displayed by the function, valid peaks are highlighted to visually indicate their significance.

The detected peaks serve as the basis for calculating the time delay between the transmission and reception of the ultrasonic pulses, ultimately leading to precise distance estimation using the speed of sound formula.

7. Detection of Real peaks due to noise interference – We have implemented a machine learning algorithm to detect real peaks in the measurements which we capture from the ultrasonic sensor, as the sensor tends to capture noise along with the real data. As shown in the figure below, while using standard libraries of python for signal processing like SciPy, the noisy peak is captured as peak value (green dot), instead of the real peak (red dot).

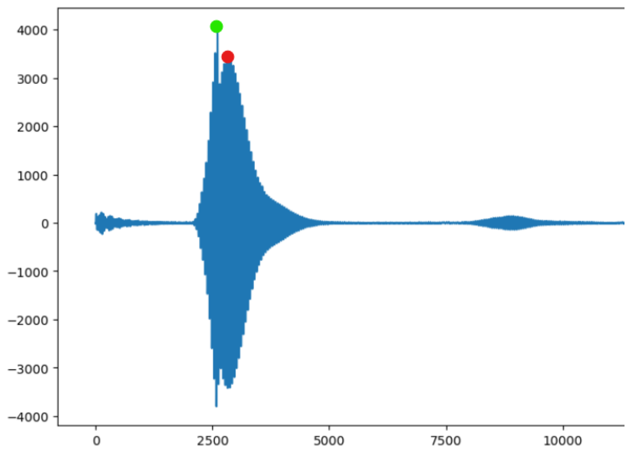


Figure 4 : Detection of False Peak due to Noise.

By combining signal processing with deep learning, for a single signal, we have split the ADC data values into multiple windows of 300 values. After that, we manually annotate the data by marking the window which has peak in it. PSD (Power spectral density) spectrogram is a very useful tool used in signal analysis. As it gives a time-signal representation of the signal, it can give an overview of how power related to different frequency components changes over time. It is also helpful in identification of noise associated with high-frequencies, and feature extraction from signals. For this reason, we took PSD spectrogram of the window values with a fixed dimension of 300 x 300 size and grayscale colormap, which has the real peak in it, as shown below on the left.

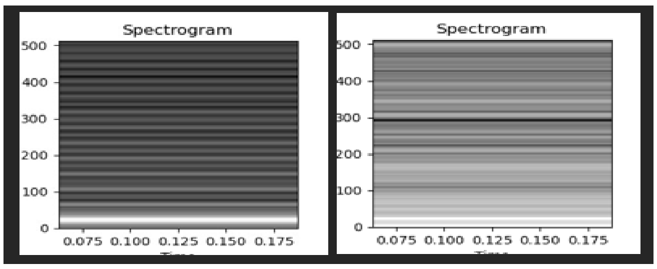
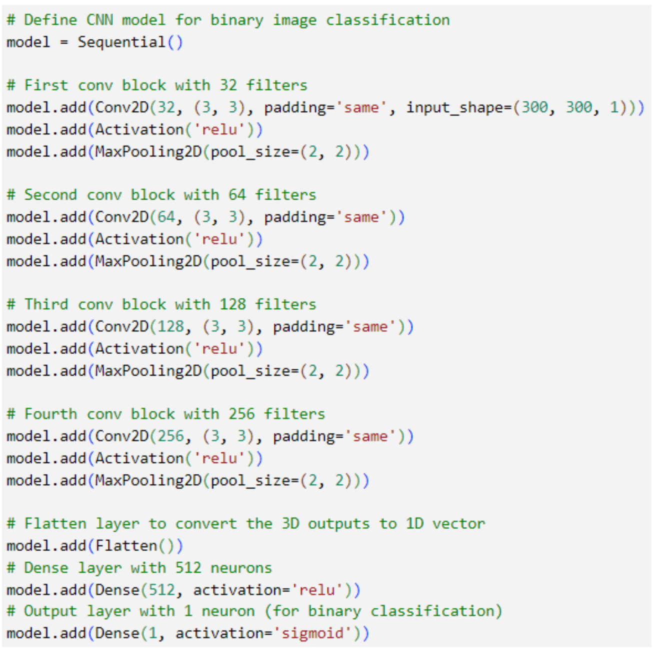


Figure 5 : spectrogram for peaks.

On the right part of the picture above, the spectrogram is of a window which does not contain the peak value. We have also taken samples of windows which do not contain peak values and taken spectrograms of them.

We have manually annotated and taken around 1600 spectrogram samples (800 for peak and 800 non-peak) covering signals from datasets from 28 cm to 200 cm distance, which has been elaborated later. Here, the distance means the distance gap between object and the sensor. We have used this data to train a Convoluted neural network (CNN) model using Keras library. PSD spectrograms can be used as input to CNN model, because it is well-suited for learning from image data. Our main objective is to classify ‘peak’ and ‘non-peak’ spectrogram images using the trained model. We have used binary classification approach to find peaks. In machine learning, binary classification is a powerful tool where the main goal is to predict any one of two possible outcomes. In our project, our images are loaded from their respective directories, converted to arrays, and then added to a list. Another list contains corresponding labels, which in our case, is 1 for peak and 0 for non-peak. We are going to convert the lists of images and labels to numpy arrays. We have normalized by dividing 255 to scale between 0 and 1. We have split the data into 80:20 ratio (80% for training and 20% for validation) using train_test_split from scikit learn library.

Our CNN model for binary classification model has four convolution blocks. Each block is followed by a ReLU activation function and a max pooling layer. After that, the model has a flatten layer, a dense layer with 512 neurons and output layer with one neuron, which will be used for binary classification.



```python
# Define CNN model for binary image classification
model = Sequential()

# First conv block with 32 filters
model.add(Conv2D(32, (3, 3), padding='same', input_shape=(300, 300, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Second conv block with 64 filters
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Third conv block with 128 filters
model.add(Conv2D(128, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Fourth conv block with 256 filters
model.add(Conv2D(256, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten layer to convert the 3D outputs to 1D vector
model.add(Flatten())
# Dense layer with 512 neurons
model.add(Dense(512, activation='relu'))
# Output layer with 1 neuron (for binary classification)
model.add(Dense(1, activation='sigmoid'))
```

Figure 6 : CNN Model

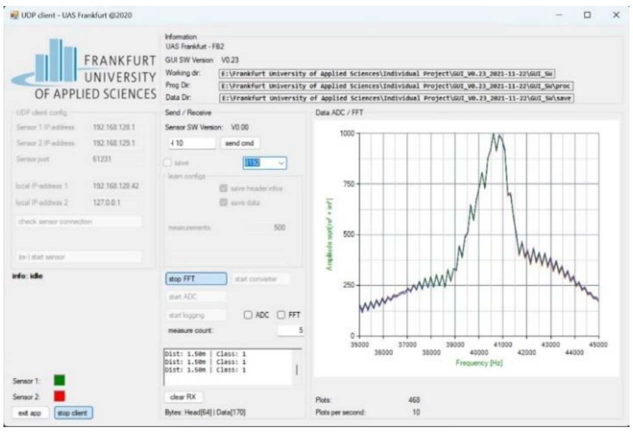
We have compiled the model with adam(Adaptive moment estimation) optimizer, with binary crossentropy as loss function and accuracy as metric. We have also defined early stopping, which will assist in stopping the training process if the validation loss does not improve after 3 epochs. We have trained the model for 10 epochs, with batch size of 64.

While testing the model, we also used the same approach (to break the test signal into multiple windows of 300 values). After that, we pass all the windows to a dictionary with corresponding window numbers as keys for identification. We are then going to generate spectrograms for all the windows and use our model to find the window which the model thinks as the most probable window that might contain a real peak, and hence from that window, derive the real peak.

IV. Experiment Results

We familiarized ourselves with the FIUS sensor and conducted experiments to gather readings. The use of the FIUS sensor allowed us to collect data for analysis. Throughout the process, we gained insights into the sensor's capabilities and functionality. This hands-on experience contributes to our understanding and further develop Machine learning algorithm and GUI for distance calculation.

1. *FUIS Sensor* - The *FUIS* sensor is equipped with user interface (GUI) application, facilitating the capture of detections through its ultrasonic sensor. It enables measurements through either Fast Fourier Transform (FFT) or Analog-to-Digital Converter (ADC) values and provides the functionality to visualize the FFT graph or ADC measurements. The FFT graph helps us to understand whether the object is either hard or smooth depending on the number of peaks, if there is a sharp peak then it would be classified as hard object. The ADC graph visualization helps us to realize how far the object is from the sensor. The stored ADC values are used for further machine learning analysis.



Information
UAS Frankfurt - FB2
GUI SW Version V0.23
Working dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW
Prog Dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW\proc
Data Dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW\save

Sensor 1 IP-address 192.168.128.1
Sensor 2 IP-address 192.168.129.1
Sensor port 61231
local IP-address 1 192.168.128.42
local IP-address 2 127.0.0.1
info: idle

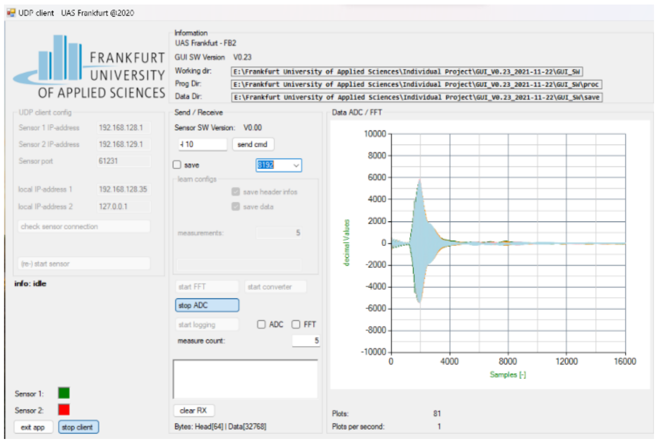
Sensor SW Version: V0.00
4 10
8192
measurements 500
measure count: 5

```
Dist: 1.50m | Class: 1
Dist: 1.50m | Class: 1
Dist: 1.50m | Class: 1
```

Bytes: Head[64] | Data[170]
Plots: 468
Plots per second: 10

Sensor 1: 🟩
Sensor 2: 🟥

Figure 7 GUI (FFT Measurement)


UDP client UAS Frankfurt @2020

Information
UAS Frankfurt - FB2
GUI SW Version V0.23
Working dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW
Prog Dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW\proc
Data Dir: E:\Frankfurt University of Applied Sciences\Individual Project\GUI_V0.23_2021-11-22\GUI_SW\save

UDP client config
Sensor 1 IP-address 192.168.128.1
Sensor 2 IP-address 192.168.129.1
Sensor port 61231
local IP-address 1 192.168.128.35
local IP-address 2 127.0.0.1
info: idle

Send / Receive
Sensor SW Version: V0.00
-I 10
save 8192
measurements: 5
measure count: 5
Bytes: Head[64] | Data[32768]

Plots: 81
Plots per second: 1

Figure 8:GUI (ADC Measurements).

a) ***Measurements (FFT)*** **-** In our experimentation, we took measurements using Fast Fourier Transform (FFT) for peak detection when dealing with hard objects. This technique allowed us to discern distinct frequency peaks associated with the characteristics of rigid materials. The precision afforded by FFT in capturing subtle frequency variations proved crucial for accurate readings in such scenarios.

b) ***Measurements (ADC)*** - \We took ADC measurements of objects placed at different distances, which are later used for computations and calculating the distance using the first peak echo detection. This technique allowed us to test and improve the accuracy of measuring the distance between the object and the sensor. In the later part, we made use of the measurements to compute the confusion matrix and improve the precision of first echo detection by searching for the maximum peak within the time window. The setup in the Laboratory is as shown in the figure below.

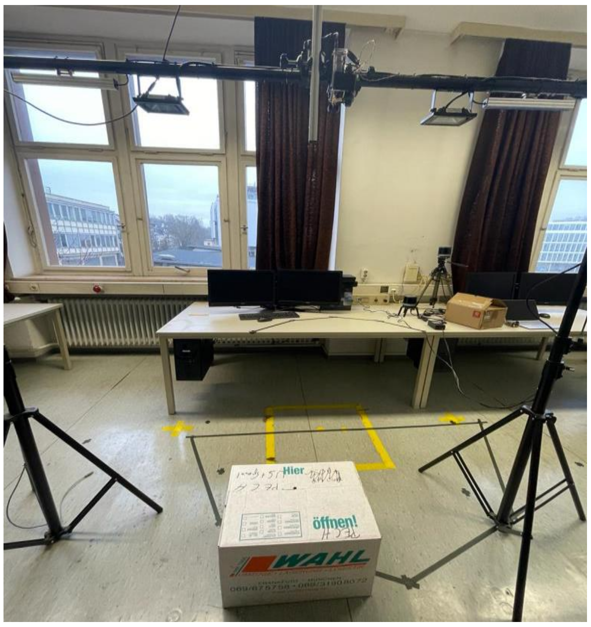


Figure 9 : FIUS sensor setup.

The figures below show the ADC measurements of object which are placed at different distances $\boldsymbol{d_{FUIS}} = \mathbf{50\ cm}$, **100 cm, 150 cm, 200 cm** respectively. In each of the instances, more than 2000 measurement readings were taken to train and test the first echo detection algorithm.

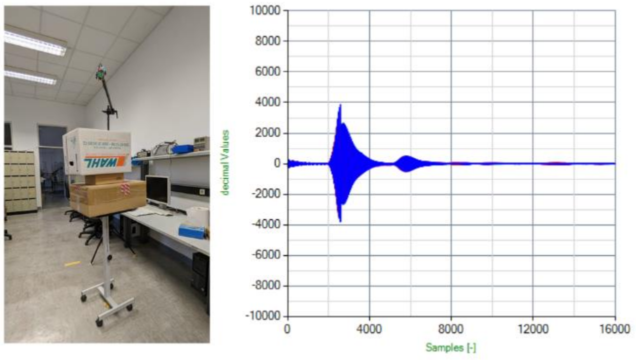


*Figure 10 Distance (**d_{man}** = 50 cm, ADC Plot)*

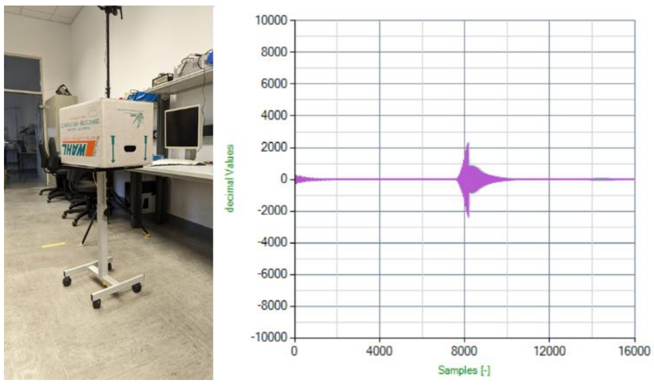


*Figure 11 : Distance Measurement Readings (**dman** = 100 cm)*

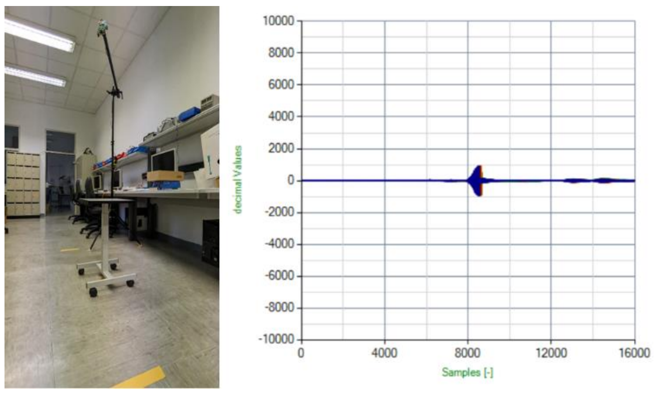


*Figure 12 Distance Measurement Readings (**dman** = 150 cm)*

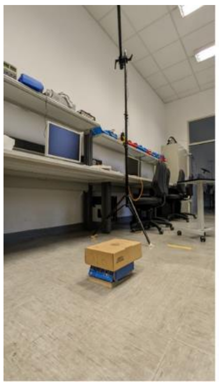
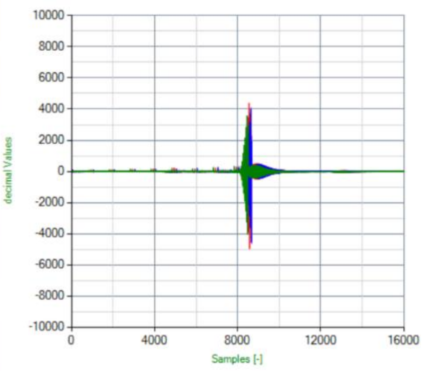
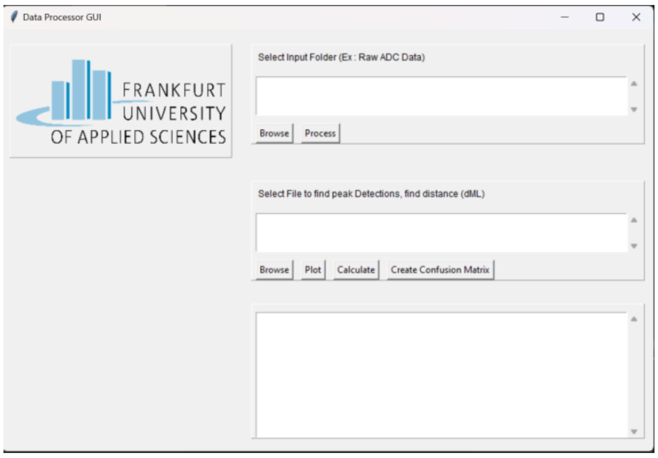



Figure 13 Distance Measurement Readings (**dman** = 200 cm)

2. *Machine Learning*

*a) **GUI Application -*** A GUI application is developed using Python in Visual Studio to utilize the measured ADC data and perform machine learning computations. The GUI enables users to select the folder containing the raw ADC data for processing. Since the data includes headers that are unnecessary for plotting ADC samples, the application processes the data and stores it in a separate folder. Users can choose any processed data to calculate the distance of the object from the sensor.



Data Processor GUI

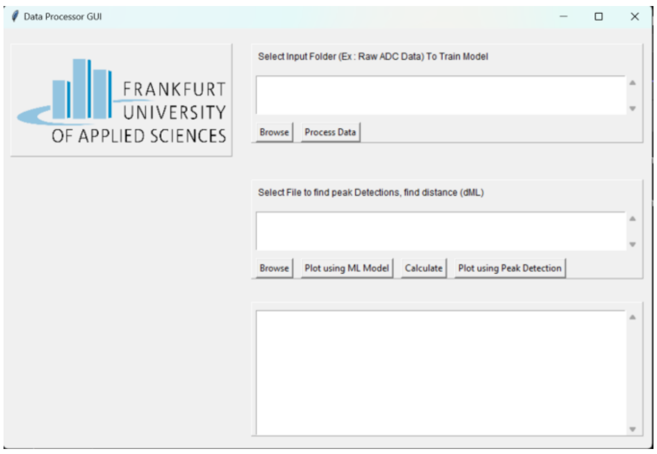
Select Input Folder (Ex : Raw ADC Data)

Browse | Process

Select File to find peak Detections, find distance (dML)

Browse | Plot | Calculate | Create Confusion Matrix

Figure 14 : ML driven GUI.



Data Processor GUI

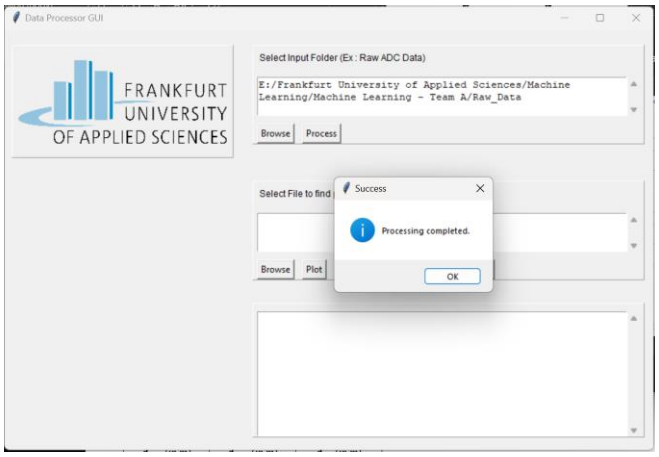
Select Input Folder (Ex : Raw ADC Data) To Train Model

Browse | Process Data

Select File to find peak Detections, find distance (dML)

Browse | Plot using ML Model | Calculate | Plot using Peak Detection

Figure 15: ML driven GUI.



Data Processor GUI

Select Input Folder (Ex : Raw ADC Data)

E:/Frankfurt University of Applied Sciences/Machine Learning/Machine Learning - Team A/Raw_Data

Browse | Process

Select File to find

Success

Processing completed.

OK

Browse | Plot

Figure 16 : GUI for Processing the Raw ADC Data

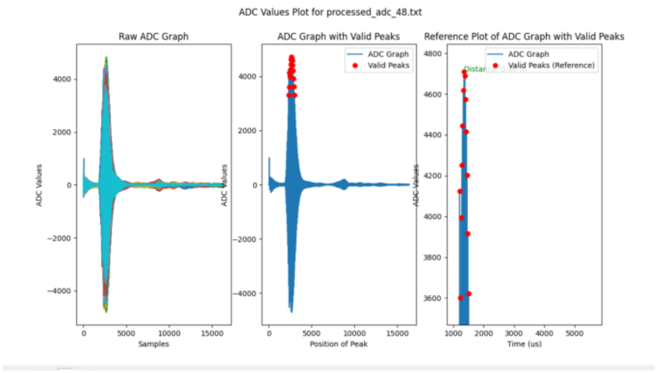
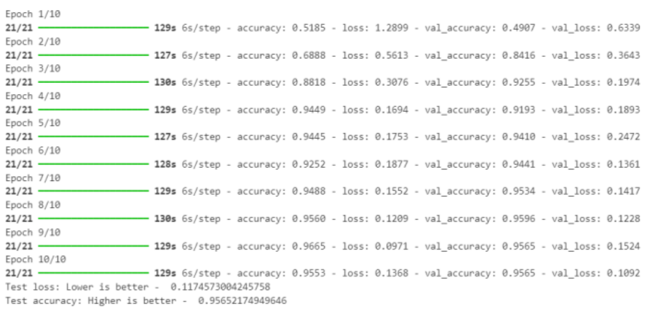


Figure 17 : Plotting and calculating dML with first echo detection.

Moreover, the GUI provides options for users to view the manually measured distance (d_{MAN}), as well as the distances calculated using machine learning algorithms (d_{ML}, d_{FIUS}). Additionally, users can generate a confusion matrix. If the deviation between the manually measured distance (d_{MAN}) and the automatically created distance (d_{ML}. d_{FIUS}) is less than 1 cm, then it shall be counted as a hit. Otherwise, it's a failure.

b) Training & Testing for First echo detection algorithm - We have used our trained CNN model to predict real peak from a test signal and the result is given below:



```
Epoch 1/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 129s 6s/step - accuracy: 0.5185 - loss: 1.2899 - val_accuracy: 0.4907 - val_loss: 0.6339
Epoch 2/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 127s 6s/step - accuracy: 0.6888 - loss: 0.5613 - val_accuracy: 0.8416 - val_loss: 0.3643
Epoch 3/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 130s 6s/step - accuracy: 0.8818 - loss: 0.3076 - val_accuracy: 0.9255 - val_loss: 0.1974
Epoch 4/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 129s 6s/step - accuracy: 0.9449 - loss: 0.1694 - val_accuracy: 0.9193 - val_loss: 0.1893
Epoch 5/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 127s 6s/step - accuracy: 0.9445 - loss: 0.1753 - val_accuracy: 0.9410 - val_loss: 0.2472
Epoch 6/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 128s 6s/step - accuracy: 0.9252 - loss: 0.1877 - val_accuracy: 0.9441 - val_loss: 0.1361
Epoch 7/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 129s 6s/step - accuracy: 0.9488 - loss: 0.1552 - val_accuracy: 0.9534 - val_loss: 0.1417
Epoch 8/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 130s 6s/step - accuracy: 0.9560 - loss: 0.1209 - val_accuracy: 0.9596 - val_loss: 0.1228
Epoch 9/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 129s 6s/step - accuracy: 0.9665 - loss: 0.0971 - val_accuracy: 0.9565 - val_loss: 0.1524
Epoch 10/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 129s 6s/step - accuracy: 0.9553 - loss: 0.1368 - val_accuracy: 0.9565 - val_loss: 0.1092
Test loss: Lower is better -  0.1174573004245758
Test accuracy: Higher is better -  0.95652174949646
```

Figure 18 : CNN model training results.

Our model achieved a training loss of 0.1175 and accuracy of 95.65%. We have also created a confusion matrix to compare the results.

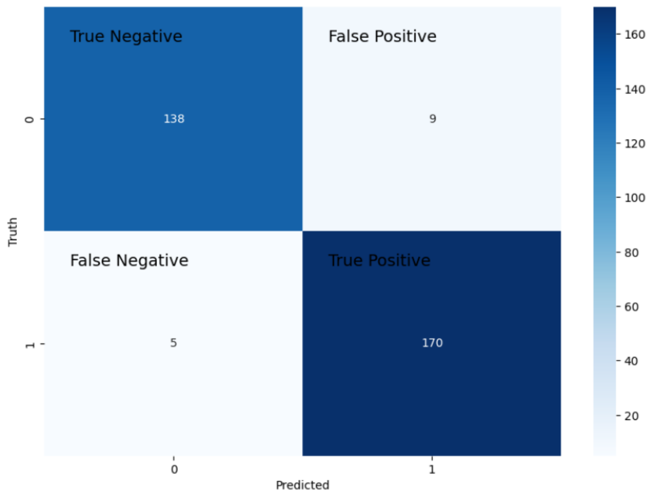
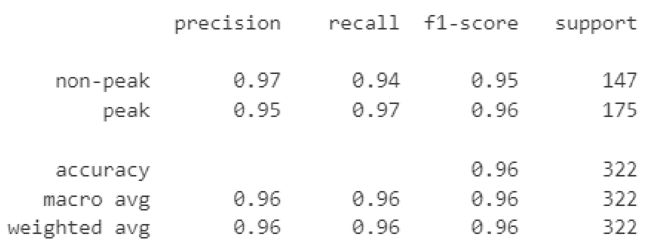


Figure 19 : Confusion Matrix



```
              precision    recall  f1-score   support

    non-peak       0.97      0.94      0.95       147
        peak       0.95      0.97      0.96       175

    accuracy                           0.96       322
   macro avg       0.96      0.96      0.96       322
weighted avg       0.96      0.96      0.96       322
```

Figure 20 : Classification Report

We have used our trained CNN model to predict real peak from a test signal and the result is given below.

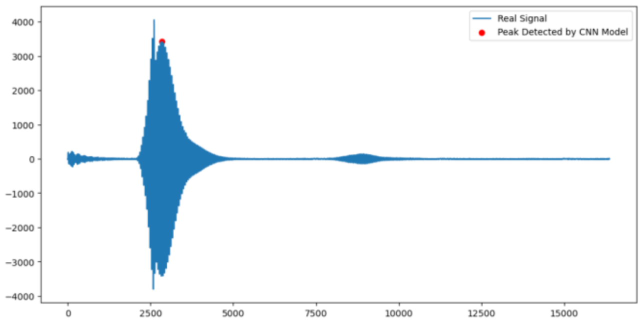


Figure 21 : Predicting Real Peak

Training with a greater number of data spectrograms will improve the performance of the CNN model, which can then be used to determine the real peak of our test signal.

Distance Measurements and Readings		
d_{Man} (in cm)	**d_{FUIS}** (in cm)	**d_{ML}** (in cm)
20	~19	23
25	~23	30
50	~53	46
100	~98	110
150	~146	170
200	~197	180

IV. Conclusion

In conclusion, we can say that, with the combined signal processing and deep learning approach, we can improve the object detection accuracy of the sensor system. However, more data covering a wider spectrum of distances from sensor to object to train CNN model will improve its accuracy. Adding more layers or tuning hyperparameters, such as learning rate, and number of epochs, can improve the CNN model accuracy. We also need to test the model on more real-world data so that its performance can be evaluated. We can also reduce noise on the source ADC signal itself by using some kind of filter for instance.

V. REFERENCES

[1] "Machine learning - Wikipedia", https://en.wikipedia.org/wiki/Machine_learning#Artificial_intellig%20ence, (Last Accessed: 11.3.2024).

[2] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 3, pp. 1–21, Mar. 2021, doi: https://doi.org/10.1007/s42979-021-00592-x.

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

[4] "Red Pitaya", Red Pitaya in Industry - Red Pitaya, (Last Accessed: 13.03.2024)

[5] "Red Pitaya",RedPitayaStarterKits.jpg (760×382) (mouser.in), (Last Accessed: 13.03.2024)

[6] "Red Pitaya (computer)," Wikipedia, Jan. 09, 2024. https://en.wikipedia.org/wiki/Red_Pitaya_(computer)#:~:text=It%20has%20three%20USB%202.0%20ports%2C%20Wi-Fi%2C%20Ethernet (Last Accessed: 13.3. 2024).

[7] C. Van Loan, "Computational Frameworks for the Fast Fourier Transform," Jan. 1992, doi: https://doi.org/10.1137/1.9781611970999.

[8] SalRite, "Demystifying 'Confusion Matrix' Confusion," Medium, Dec. 18, 2018. https://towardsdatascience.com/demystifying-confusion-matrix-confusion-9e82201592fd, (Last Accessed: 14.3. 2024).

[9] Geeksforgeeks, "Confusion Matrix in Machine Learning - GeeksforGeeks," GeeksforGeeks, Feb. 07, 2018. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (Last Accessed: 14.3. 2024).

[10] S. Bansal, "Supervised and Unsupervised learning - GeeksforGeeks," GeeksforGeeks, Apr. 17, 2019. https://www.geeksforgeeks.org/supervised-unsupervised-learning/, (Last Accessed: 15.3. 2024).

[11] N. Bustos, Mehrsa Mashhadi, S. K. Lai-Yuen, S. Sarkar, and T. K. Das, "A systematic literature review on object detection using near infrared and thermal images," Neurocomputing, vol. 560, pp. 126804–126804, Dec. 2023, doi: https://doi.org/10.1016/j.neucom.2023.126804.

[12] T. Nesti, Santhosh Boddana, and B. Yaman, "Ultra-Sonic Sensor based Object Detection for Autonomous Vehicles," Jun. 2023, doi: https://doi.org/10.1109/cvprw59228.2023.00026.

[13] H., Liu, C. Wu, & H, Wang, "Real time object detection using LiDAR and camera fusion for autonomous driving", Sci Rep 13, 8056 (2023). https://doi.org/10.1038/s41598-023-35170-z

[14] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An Efficient Probabilistic 3D Object Detector

for Autonomous Driving," IEEE Xplore, Jun. 01, 2019. https://doi.org/10.1109/CVPR.2019.01296.