Parameter Setting and Reliability Test of a Sensor System for Baby Detection in a Baby Seat in a Car

M. Engg. In Information Technology
Autonomous Intelligent Systems and
Machine Learning

By Dr. Peter Nauth and Andreas Pech

Kushal Prakash
1429800
Kushal.prakash@stud.fra-uas.de

Madhushree manjunatha Lakshmidevi
1445185
Madhushree.manjunatha-laskhmidevi@stud.fra-uas.de

***Abstract*—The detection and recognition of baby in car seats pose a critical challenge in ensuring their safety during travel. Existing solutions often lack the necessary precision required for reliable detection. We propose leveraging advanced machine learning algorithms to address this. In our study, we recorded data from ultra sonic sensor under various conditions and conducted experiments to detect baby seated in baby seat in a car seat, especially under challenging conditions such as the baby seat covered by sunscreen cloth. We employed MultiLayer Perceptron(MLP) and Convolutional Neural Networks (CNN) for supervised learning to train and test our model. To check its efficacy against other models, we also trained and tested Support Vector Machines (SVM) model for comparing. By evaluating accuracy and F1 scores through confusion matrices we realised thresholding plays an important role in accurate results, we assessed the effectiveness of our approach in different scenarios, including detecting an empty baby seat in a car versus a baby seat occupied by a baby with different obstacles.**

Keywords— Red Pitaya, FIUS, Ultrasonic sensor SRF02, Machine learning, Supervised learning, Convolution Neural Network, confusion matrix.

I. INTRODUCTION

Our research is centred on Ultrasonic sensors employing sound waves to detect whether a baby is in a car seat. These sensors release ultrasonic waves that travel in the direction of the car seat. When these waves come into contact with an item, in our case a baby in a car seat, they are reflected back towards the sensor [1]. By measuring the time it takes for the ultrasonic waves to travel to the object and back, the sensor calculates the distance to the object. This distance information is then analysed to determine if an object is present in the car seat [2]. This time to return also depends on texture, shape and other parameters. Usually, the device is set up with a threshold distance, and when the measured distance is within that range, it means the baby is there. The integration of this detection technique with automotive safety systems enables the installation of warnings or notifications to caregivers in the event that the baby's presence is identified, thereby augmenting vehicle safety and monitoring[3].

To achieve this, we take the Digital(ADC) data and convert it to (Fast Fourier Transform)FFT and use this data to train the machine learning algorithms mentioned in the abstract. Furthermore, we optimise these networks using thresholding concept to increase accuracy and reduce false classification.

II. METHODOLOGY

In this section we will further discuss on Red Pitaya Ultrasonic sensor, conversion of data to FFT for analysis, machine learning algorithm backdrop, and confusion matrix[4].

A. Ultrasonic sensor and the Red Pitaya measurement board

The Red Pitaya STEM Lab is a test and measurement board that utilises a System-on-a-Chip (SoC) that was previously manufactured by Xilinx. It can be remotely operated and set up to operate as an (Inductance(L), Capacitance(C) and Resistance(R))LCR meter, network analyser, oscilloscope or spectrum analyser. In this setup, the Ultrasonic Sensor SRF02 a Single transducer ultrasonic RangeFinder in a small footprint PCB was used. As the SRF02 uses a single transducer for both transmission and reception, its minimum range is longer than that of other dual transducer rangers.

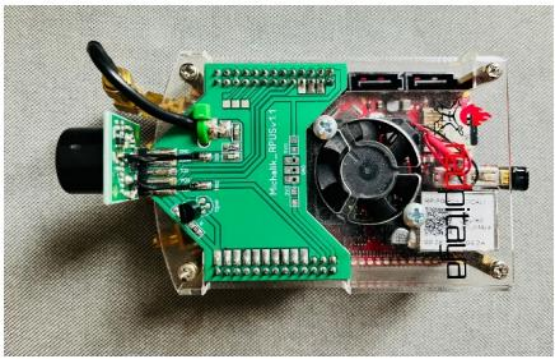

Fig. 1: Ultrasonic sensor and Redpitaya device

The smallest measurement range is about six inches or 15 cms. It can run on a 5V grounded power supply. Data can be wirelessly sent to a laptop/computer for further processing thanks to the WiFi capabilities in Red Pitaya using the general purpose USB ports. GNU/Linux is the operating system that powers the sensor. They can be used to organise and store measurements on a PC or mobile device. Two analog RF inputs and outputs are included in the main Red Pitaya unit in addition to 16 normal input and output ports. The board also has a micro-SD card slot, an Ethernet RJ45 connection, a USB port, and a micro-USB connector for the console. The device operates in the 50 MHz frequency spectrum.

B. Measuring methods using FIUS sensor

A UDP client was provided by the Frankfurt University of Applied Sciences which helps us to interact with the FIUS sensor. Upon starting the device and the tool, we ran the sensor to collect ADC and FFT data in different instances to check the capabilities of the tool and the device. This helped us in understanding the device behaviour and the tool usage. In the below images we have listed some of the objects which we tested the sensor against to get the feel of the tool.

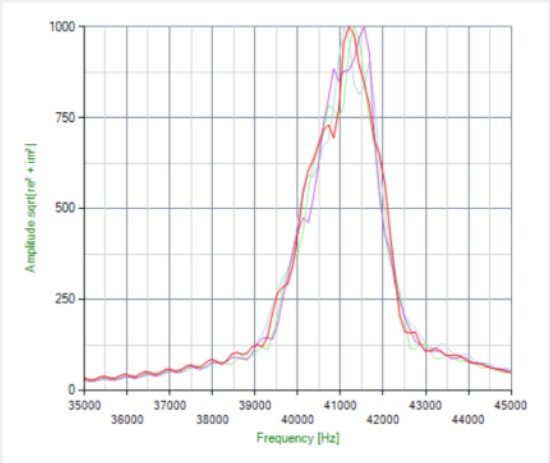

Fig. 2: FFT of the rough surface (wall)

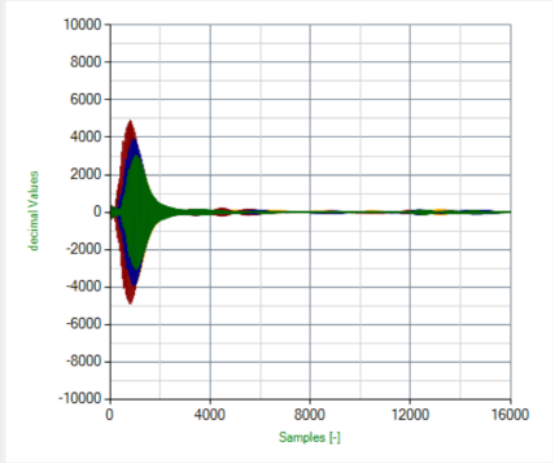

Fig. 3: ADC of the rough surface (wall)

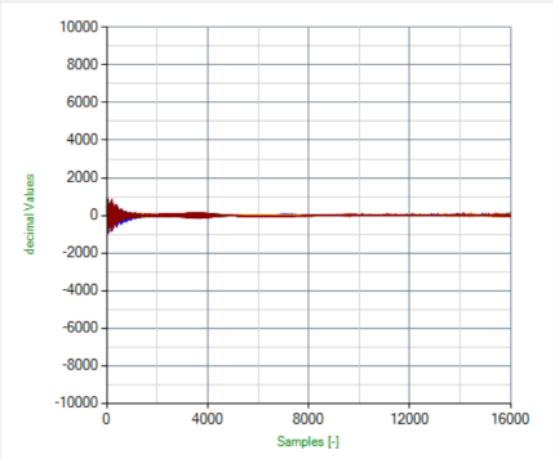

Fig. 4: ADC of the soft surface (human)

C. Fast Fourier Transform

The Fast Fourier Transform (FFT) is an efficient algorithm used for computing the Discrete Fourier Transform (DFT) of a sequence. The DFT operation converts a signal from the time-domain into its corresponding representation in the frequency domain. As our goal was to collect the data in ADC form and then convert it to FFT.

Utilising a divide-and-conquer approach, the FFT algorithm combines the results in O(N) time complexity by recursively splitting a DFT of size N into two DFTs of size N/2. This results in significantly faster computation compared to the naive O(N^2) technique.

The FFT technique is widely used in many sectors where the analysis or modification of signals in the frequency domain is essential, including signal processing, data compression, picture processing, and many more. It is now a common tool used in many technical and scientific fields [5].

This transformation is shown in the formula below.
There are three different dominant frequencies in the signal across the duration that was recorded.

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

One such tool is the Fast-Fourier Transform (FFT). This enables the Fourier Transform to be applied to actual data. Spectral analysis, computational physics, and audio/video production are some examples of applications.

FFT is limited in the amount of information it can extract from a signal, despite its significant contributions to science and technology. The Fourier Transform specifically reveals global frequency data. [6]
We wrote our own code to do the same. This helped us reduce the size and time needed for model training and testing as the final data was stored to numpy array format.

D. Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification or machine learning algorithm. By comparing the predicted labels of a collection of data to the actual labels, a confusion matrix table that summarises the performance of a classification method is obtained. An example of confusion matrix is shown in Fig. 5.

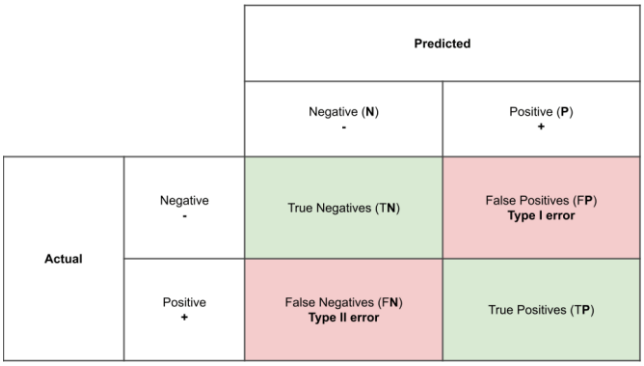

Fig. 5: Confusion Matrix

The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are:

1. *TP (True Positive):* When the expected value matches the actual value, we have a true positive case. This indicates that the model predicted a positive result, and the actual result is positive.
2. *TN (True Negative):* When the expected value and the real value are the same, a truly negative situation arises. This time, however, the model predicted a negative value, and the actual value is also negative.
3. *FP (False Positive):* When the expected value matches with an incorrect value, it known as a false positive. The model anticipated a favourable outcome even though the actual value was negative. This is called the type I prediction error.
4. *FN (False Negative):* When the expected value matches with an incorrect value, this is referred to as a false negative case. Even though the actual result was positive, the model predicted a negative result. This is the type II prediction error.

These four parameters are used to obtain other performance metrics which can be used to better judge the model.

Some additional performance metrics of an algorithm are accuracy, precision, recall, and F1 score, which are calculated on the basis of the above-stated TP, TN, FP, and FN[7].

A number of significant metrics, including the following ones, can be computed using the confusion matrix to assess how well the classification algorithm performed:
Accuracy: The ratio of correctly predicted samples to all samples in the dataset is known as accuracy. It calculates the frequency with which the classifier predicts the right thing. The formula for accuracy is:

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FP+FN+TN)} \qquad \ldots(2)$$

Precision: The ratio of accurately predicted positive samples to the total number of positive samples the model projected is known as precision. It counts the proportion of expectedly positive samples that really are positive. The precision formula is:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \ldots(3)$$

Recall (Sensitivity): The ratio of accurately predicted positive samples to all positive samples in the dataset is known as recall. It calculates the proportion of real positive samples that the model accurately predicted. The recall formula is as follows:

$$\text{Recall} = \frac{TP}{TP+FN} \qquad \ldots.(4)$$

F1 Score: The harmonic mean of recall and precision is the F1 score. When recall and precision are both significant, it is a useful metric. The F1 score formula is:

$$\text{F1Score} = \frac{2*precision*recall}{precision+recall} \qquad \ldots.(5)$$

Recognising the widespread emphasis on accuracy when assessing performance is crucial before delving into confusion matrices, the foundation of model evaluation. Though it has its uses, precision by itself is not always reliable. Confusion matrices' applications and the drawbacks of accuracy as a stand-alone statistic will both be discussed in this section.

- *True Positive Rate (TPR):* Also known as recall, this metric represents the proportion of positive cases the model identified correctly (TP / (TP + FN)).

- *False Positive Rate (FPR):* This metric represents the proportion of negative cases the model incorrectly classified as positive (FP / (TN + FP)).
- *True Negative Rate (TNR):* Also known as specificity, this metric represents the proportion of negative cases the model identified correctly (TN / (TN + FP)).
- *False Negative Rate (FNR):* This metric represents the proportion of positive cases the model incorrectly classified as negative (FN / (TP + FN)).

Depending on the issue and application, these rates can be helpful for assessing different parts of a classification model's performance. The model's ability to correctly identify positive cases is measured by TPR, while its inclined to incorrectly classify negative circumstances as positive is measured by FPR. The model's capacity to accurately identify negative scenarios is measured by TNR, while its inclined to incorrectly classify positive events as negative is measured by FNR. To create a high-performing classification model, we generally wish to maximise TPR and TNR while reducing FPR and FNR[8].

F. Model training

Machine learning, a cornerstone of artificial intelligence, delves into creating algorithms and models that equip computers with the ability to learn from data. This enables them to make predictions or judgments without relying on explicit programming. In essence, machine learning empowers computers to learn from examples and experience, shifting away from dependence on pre-defined rules or constant human intervention.

The process of providing data to an ML algorithm in order to assist in identifying and teaching it suitable values for all related attributes is known as model training in machine language. While there are many other kinds of machine learning models, supervised and unsupervised learning are the most widely used.

When both the input and output values are included in the training data, supervised learning can take place. A supervisory signal is any piece of data that has both the inputs and the anticipated result. When inputs are given into the model, the training process is dependent on how much the processed result deviates from the documented outcome.

1. *Multi Layer Perceptron (MLP):*

A multilayer perceptron (MLP) represents a type of feedforward neural network composed of fully connected neuron's employing nonlinear activation functions. It finds extensive application in discerning data that isn't linearly separable.

MLPs enjoy widespread usage across diverse domains, including image recognition, natural language processing, and speech recognition, among others. Their architectural flexibility and capability to approximate any function given specific conditions render them pivotal in deep learning and neural network research [12].

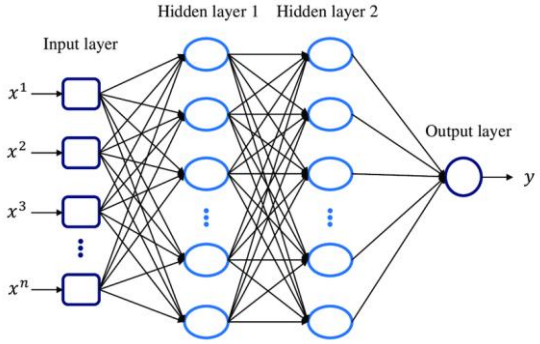

Fig. 6: Multi Layer Perceptron architecture

2. *Convolutional Neural Network (CNN)*

A Convolutional Neural Network (CNN) represents a specialised form of deep learning algorithm tailored for tasks involving image recognition and processing. Composing various layers such as convolutional, pooling, and fully connected layers, CNNs mimic the visual processing mechanism of the human brain. Their architecture is adept at discerning hierarchical patterns and spatial relationships within images [9].

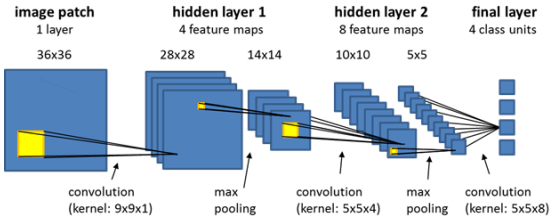

Fig 7: Convolutional Neural Network architecture

3. *Support Vector Machine (SVM):*

Support Vector Machine (SVM) stands out as one of the most widely employed Supervised Learning algorithms, serving both Classification and Regression tasks. Nevertheless, its principal usage remains in Classification problems within Machine Learning.

The primary objective of the SVM algorithm is to construct the optimal line or decision boundary capable of segregating an n-dimensional space into distinct classes. This facilitates the straightforward categorisation of new data points into their respective classes in subsequent instances. Termed as a hyperplane, this optimal decision boundary defines the separation between classes [10].

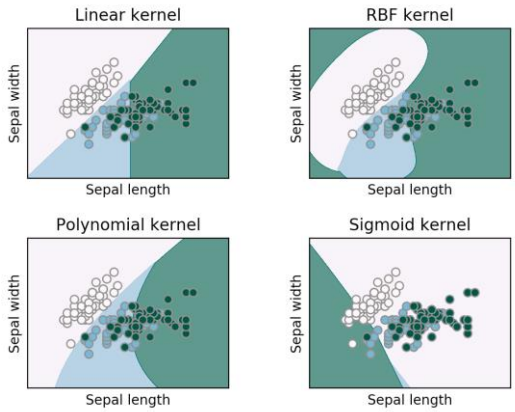

Fig 8: SVM with linear and nonlinear kernel

The following steps are described to create a MLP, CNN and SVM models to detect a baby presence or absence in the baby carriage on the passenger seat inside the car:

1. *Data Preparation:* Load the dataset and divide it into training and testing sets. Preprocess the data, including scaling, normalisation.
2. *Model Creation:*

- *For MLP:* Instantiate the MLP model defining the number of hidden layers and maximum number of iteration and activation functions.
- *For CNN:* Design the architecture by specifying convolutional layers, pooling layers and fully connected layers.
- *For SVM:* Instantiate the SVM model, such as using LinearSVC, and set hyper parameters like C.

3. *Model Training: For MLP, CNN and SVM train the model on the training data using the fit() method.*

4. *Model Evaluation: For MLP, CNN and SVM predict labels of test data and evaluate performance using metrics like accuracy, precision, and recall.*

Finally, machine learning presents a strong method for endowing computers with the capacity to learn from data and make wise decisions. We can unleash the power of image/object analysis for tasks like object recognition and classification by utilising supervised learning algorithms such as MLPs, CNNs, and SVMs. The choice depends on the particular problem and the properties of the data, as each technique has advantages of its own.

III. IMPLEMENTATION

A. *Experimental setup and sensor placement*

This experiment utilises a Ford Fiesta v16 model in white colour as the test subject. The car is parked in a controlled environment a basement garage at Frankfurt University of Applied Sciences to ensure consistent measurement conditions.

Fig. 9: Ford Fiesta v16 Model used for measurement

For data acquisition, a Red Pitaya measurement board equipped with an ultrasonic sensor is mounted on the passenger side dashboard. The sensor's placement is directly in front of the passenger seat, facing the passenger's side.

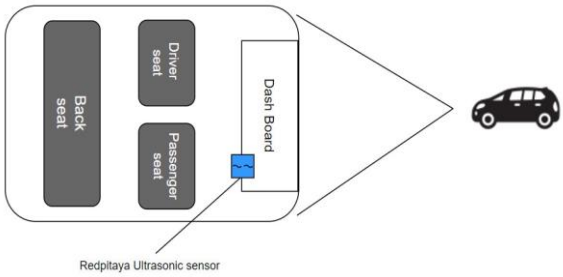

Fig. 10: Top view diagram of the car

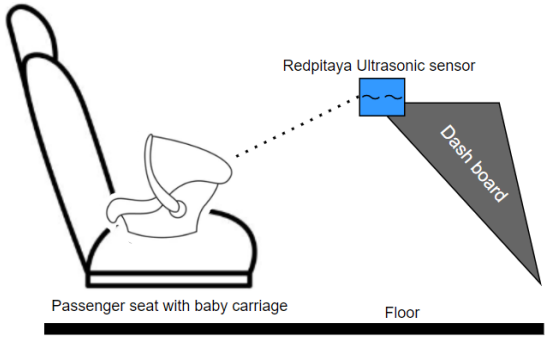

Fig. 11: Passenger-seat with baby carriage diagram view (side view)

B. *Measurements*

We conducted essential measurements to assess various aspects, including the baby's height, distances from sensors to the seat, carriage, baby, and door. The baby's total length was measured to be 42 centimetres, providing insight into their size within the environment.

Fig 12: Measuring the baby's height

The direct distance from the sensor to the door was recorded as 41 centimetres. Additionally, the measurement of the farthest point from the sensor to the door was found to be 81 centimetres, indicating the maximum distance within this relationship.

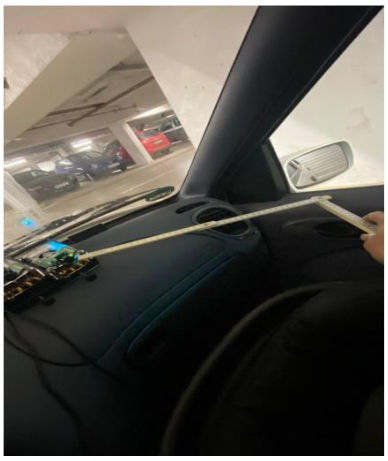

Fig. 13: Sensor to door measurement

The distance between the sensor and the carriage, which is likely the stroller or another form of transportation for the baby, was determined to be 23 centimetres.

Fig. 14: Sensor to carriage measurement

A critical measurement, the distance between the sensor and the baby's head, was recorded at 43 centimetres, providing information on the proximity of the monitoring device to the baby's vital area.

Fig. 15: Measurement from sensor to baby's head

The distance between the sensor and the seat, likely a reference to the baby's seating arrangement, measured 80 centimetres, indicating the spatial relationship between the sensor and the designated seating area.

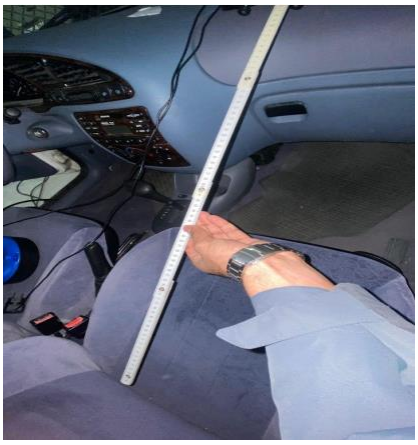

Fig. 16: Sensor to seat measurement without baby carriage

C. Data Collection

Utilising Red Pitaya technology, the ADC dataset was acquired from the car with the baby seat, comprising 400 samples each time. These ADC sensors positioned within the vehicle becomes an indispensable aspect of the vehicle's monitoring system. With the assistance of Red Pitaya, these sophisticated sensors meticulously capture a diverse array of analog signals, providing insight into the nuanced dynamics within the car's interior environment. Spanning parameters such as vibration intensities, temperature fluctuations, and ambient sound levels, the dataset offers a holistic view of the ecosystem surrounding the baby seat. Through the data acquisition facilitated by Red Pitaya, these sensors record even the slightest of variations, demonstrating a commitment to precision and reliability. During this data collection, we have changed the position of the baby by tilting its head, adjusting its position, changing hands position, adding a feeding bottle to its hand, covering the sunscreen, adding blanket and adding scarf. We have also collected data by varying the carriage's position. In total, we have collected 1,16,882 unique data samples. Out of which 55,883 are carriage without baby and 36,599 with baby and 24,400 with baby having some obstacle like scarf, sunscreen and blanket.

D. Data Format

For this measurement, we used CSV file data that was exported from the software. Each dataset has 16,400 columns. Out of which first 16 columns are the header data and rest are the ADC data.

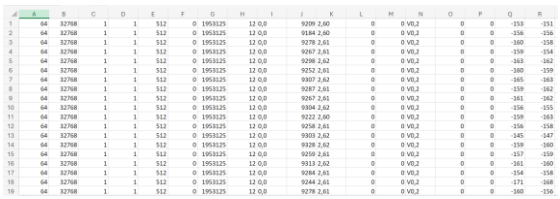

Fig. 17: Raw measurement data

IV. MACHINE LEARNING ALGORITHMS

1. Data Pre-processing:

After extracting data from the Red-Pitaya measurement board along with the ultrasonic sensor is initially stored in .csv format. Python's Pandas and NumPy modules are employed to transform the data into *.npy files(NumPy array format) for further processing as it is very lightweight and uses less resource. Subsequently, the dataset undergoes conversion from ADC data to FFT data. The dataset is then divided into training and testing sets for MLP model evaluation. Approximately 47% of the data is allocated for training the model, while the remaining 53% is set aside for assessing the model's performance during testing and prediction stages.

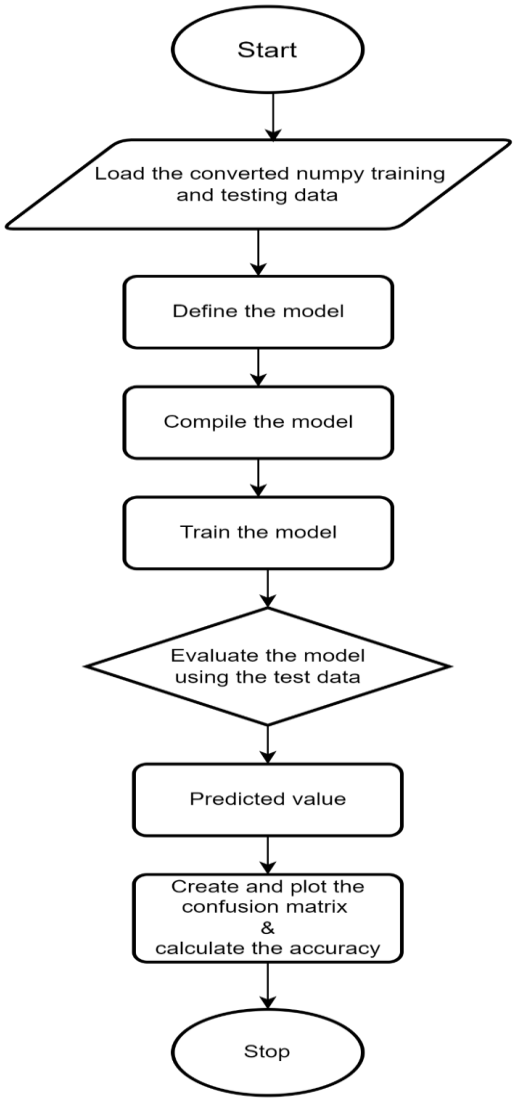

Fig. 18: Flow chart for ML models

A. Multi-Layer Perceptron model

The methodology for creating an MLP (Multi-Layer Perceptron) model to detect a baby's presence in the baby carriage on the passenger seat inside a car is implemented as illustrated in the following figure, depicting the flow chart for

data pre-processing, creation, training, and prediction of the MLP model for the project.

```
#import all the necessary packages
import numpy as np
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt2
import itertools
from joblib import dump, load
```

Fig 19: Python modules used for MLP model

The code snippet demonstrates the construction and evaluation of an MLP (Multi-Layer Perceptron) model using scikit-learn's `MLPClassifier`. `MLPClassifier` is imported from the `sklearn.neural_network` module and used to define the MLP model. Matplotlib is employed for plotting confusion matrices, while NumPy is used for loading data from .npy files. Additionally, scikit-learn's `metrics` module is utilised for generating confusion matrices to evaluate the model's performance.

```
# Define the MLP model
print("Start MLP model....")
mlp_model = MLPClassifier(hidden_layer_sizes=(256,), max_iter=1000000, activation='relu')
```

Fig. 20: Defining the MLP model

The model is defined with one hidden layer containing 256 neuron's, 'relu' activation function and the maximum number of iterations is set to 10,00,000. This optimiser, adam, is a popular choice for training neural networks and is known for its efficiency and effectiveness in handling various types of data. The model is initialised with the "adam" optimiser by default, which is a variant of Stochastic Gradient Descent (SGD) that adapts the learning rate during training. However, this is not explicitly mentioned in the code snippet.

The 'fit' method of the 'MLPClassifier' class is used to train the model on the training data 'train_data' with corresponding labels 'train_label'. However, in the given code, this line is commented out, likely because the model has already been trained and saved previously, and the saved model is loaded for evaluation.

```
# Train the MLP model
print("Training the model ...")
#mlp_model.fit(train_data, train_label)
print("")
```

Fig. 21: Train the MLP model

The test data uses the trained MLP model ('mlp_model') and stores them in 'test_pred_probs'. Then, it converts these probabilities to binary predictions using a predefined threshold ('optimal_threshold'). The predictions are stored in 'test_pred'.

```
# Convert probabilities to binary predictions using optimal_threshold
test_pred_probs1 = mlp_model.predict_proba(test_data)[:, 1]  # Probabi
optimal_threshold = 0.4  # You can adjust this threshold if needed
test_pred = (test_pred_probs1 > optimal_threshold).astype(int)
```

Fig. 22: Evaluation of the MLP model

The code demonstrates a potential form of indirect fine-tuning by adjusting the prediction threshold ('optimal_threshold'). This threshold determines how the model classifies data points based on the predicted probabilities.

B. CNN model

The methodology for creating a CNN model to detect a baby's presence in the baby carriage on the passenger seat inside a car is implemented as illustrated in the Fig. 18.
After fetching the training and testing data in NumPy array format, we feed it to CNN model.

```
#Packages needed for the model
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt2
import itertools
import tensorflow as tf
import numpy as np
from sklearn.metrics import confusion_matrix
```

Fig 23: Python modules used for CNN model

TensorFlow is utilized in constructing the CNN model, with Keras serving as the high-level TensorFlow API. The official high-level TensorFlow deep learning model construction API is known as Keras, and it is imported into the model as "tf.keras." Matplotlib is used for plotting confusion matrices, TensorFlow is utilized for creating the CNN model, NumPy is employed for reading data from the npy training and testing files, and sklearn is used to generate confusion matrices for model statistics.

```
# Define the model
print("Load CNN model......")
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(3,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Fig. 24: Defining the CNN model

The three-layer CNN model is defined using the tf.keras.Sequential approach, comprising two layers of "relu" activation with 64 and 32 nodes respectively and one layer of "sigmoid" activation.

```
# Train the model
print("Training the model with same data 10 times......")
model.fit(train_data, train_label, epochs=10, batch_size=32)
```

Fig. 25: Train the CNN model

Additionally, a confusion matrix is generated using the test class and prediction class data for the CNN model.

```
# Evaluate the model on the test data
print("Test the model......")
y_pred_prob = model.predict(test_data)
y_pred_classes = np.round(y_pred_prob)
print("------------------------------------------------------------")
print("------------------------------------------------------------")
```

Fig. 26: Evaluation of the CNN model

A thresholding feature is added to the model with the aid of previously generated confusion matrix metrics in order to fine-tune the model and increase its scores. The confusion matrix results are examined with multiple test cases in the Results and Discussion phase of the CNN model analysis. There, confusion matrix parameters including accuracy, precision, recall, and F1 Score are reviewed for both models.

C. *SVM model*

The methodology for creating a SVM model to detect a baby's presence in the baby carriage on the passenger seat inside a car is implemented as illustrated in the Fig. 18.

```
#import all the necessary packages
import numpy as np
from sklearn import metrics
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt2
import itertools
from joblib import dump, load
```

Fig 27: Python modules used for SVM model

The SVM model is also constructed using scikit-learn's SVM implementation. After loading the preprocessed NumPy training data, the SVM model is trained.

```
# Define the SVM model with a linear kernel
print("Start SVM model....")
svm_model = SVC(kernel='rbf', probability=True)
```

Fig. 28: Defining the SVM model

A confusion matrix is generated using the test class and prediction class data for the SVM model. Additionally, a thresholding feature is added to the model with the aid of previously generated confusion matrix metrics in order to fine-tune the model and increase its scores.

```
# Train the SVM model
print("Training the model ...")
#svm_model.fit(train_data, train_label)
print("")
```

Fig. 29: Train the SVM model

The confusion matrix results are examined with multiple test cases in the Results and Discussion phase of the SVM model analysis. There, confusion matrix parameters including accuracy, precision, recall, and F1 Score are reviewed for both models.

```
# Find the threshold that maximizes TPR while keeping FPR constant
fpr, tpr, thresholds = metrics.roc_curve(test_label, test_pred_probs[:,1])
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
```

Fig. 30: Evaluation of the SVM model

V. RESULTS AND CONCLUSION

The confusion matrix and its metrics, which were obtained from the trained MLP model for the test data, are described in this section. The test data scenario comprises of various scenarios of baby in the baby seat. This incident is regarded as a test run for the model. By varying the orientation, position, and addition of components, several test situations are investigated.

For testing purposes, a total of 61,204 FFT data points are considered from the dataset, representing approximately 53% of the overall data. The remaining 47% of the data is allocated for training both the MLP, CNN and SVM models.

A. *MLP model*

Consequently, two Confusion matrices are generated for each case. The first confusion matrix represents the results without the thresholding feature, while the second confusion matrix reflects the outcomes with the thresholding feature. Without thresholding the classification is as follows, TPR and TNR are 0.69 and 0.89 respectively, while the accuracy and precision was 76.42% and 0.93 respectively. And after employing threshold the True Positive Rate (TPR) and True Negative Rate (TNR) stands at 90.76% and 86.06%, respectively. The accuracy 89.21% and precision 92.96%. These rates indicate the accuracy of predicting a baby in a baby seat, in the MLP model. Furthermore, there is a False Negative Rate (FNR) of 9.23% and a False Positive Rate (FPR) of 13.93%, signifying the occurrence of false positive and negative results in the MLP model.

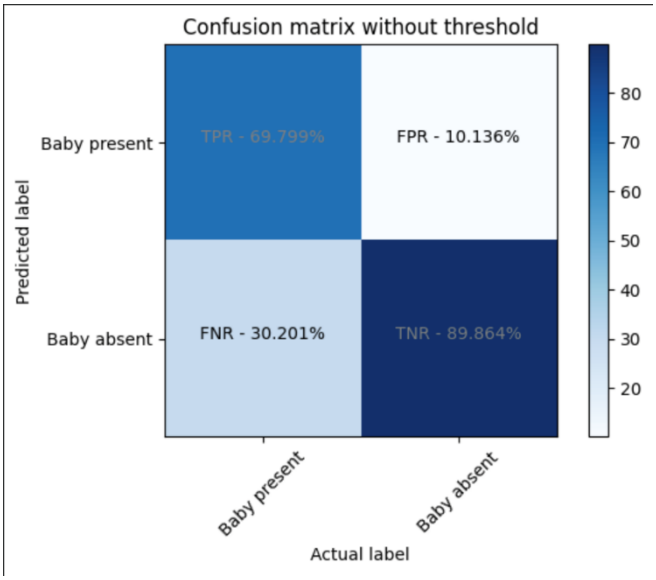

Fig 23: Confusion Matrix without threshold for MLP model

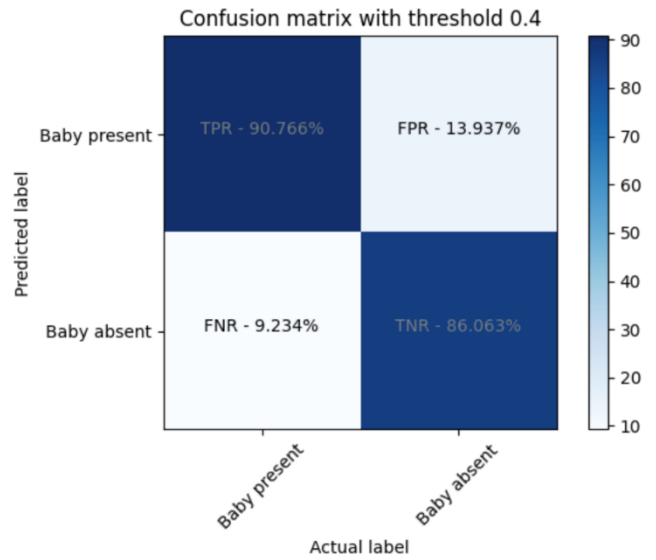

Fig 24: Confusion Matrix with threshold for MLP model

B. CNN model

For the CNN model, two Confusion matrices are generated for each case. The initial matrix presents results without the dynamic thresholding feature, while the subsequent matrix depicts outcomes incorporating the dynamic thresholding feature. Without thresholding the classification is as follows. TPR and TNR are 0.92 and 0.45 respectively, while the accuracy and precision was 76.55% an 77.27% respectively. And after employing threshold the True Positive Rate (TPR) and True Negative Rate (TNR) are observed at 97.546% and 45.048%, respectively. These rates indicate the accuracy of predicting a baby in a baby seat, in the CNN model. Additionally, a False Negative Rate (FNR) of 2.454% and a False Positive Rate (FPR) of 54.952% indicate occurrences of false negative and false positive results within the CNN model.

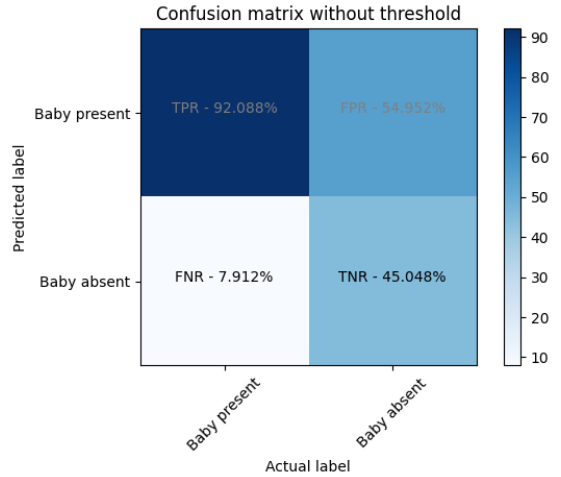

Fig 25: Confusion matrix without threshold of CNN model

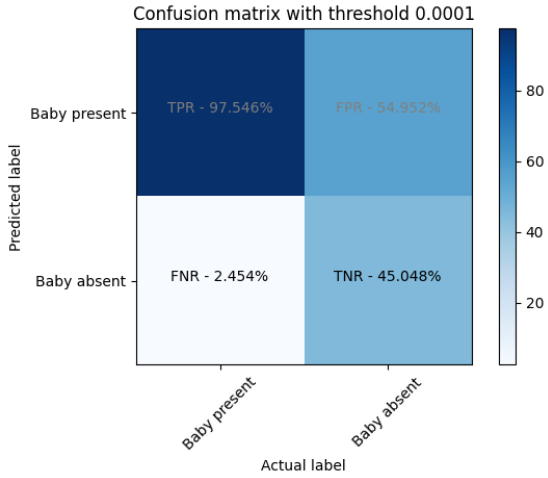

Fig 26: Confusion matrix with threshold of CNN model

C. SVM model

For the SVM model, two Confusion matrices are generated for each case. The initial matrix presents results without the dynamic thresholding feature, while the subsequent matrix depicts outcomes incorporating the dynamic thresholding feature. Without thresholding the classification is very bad as TPR and TNR are 0.0 and 1.0 respectively, while the accuracy and precision was 33.01% and 1.0 respectively. And after employing threshold the True Positive Rate (TPR) and True Negative Rate (TNR) are observed at 97.546% and 45.048%, respectively. The accuracy and precision were 81.15% and 78.2% respectively. These rates indicate the accuracy of predicting a baby in a baby seat, in the SVM model. Additionally, a False Negative Rate (FNR) of 2.454% and a False Positive Rate (FPR) of 54.952% indicate occurrences of false negative and false positive results within the SVM model.

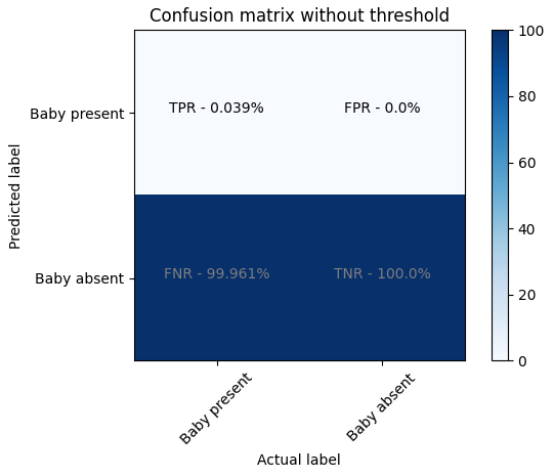

Fig 27: Confusion matrix without threshold of SVM model

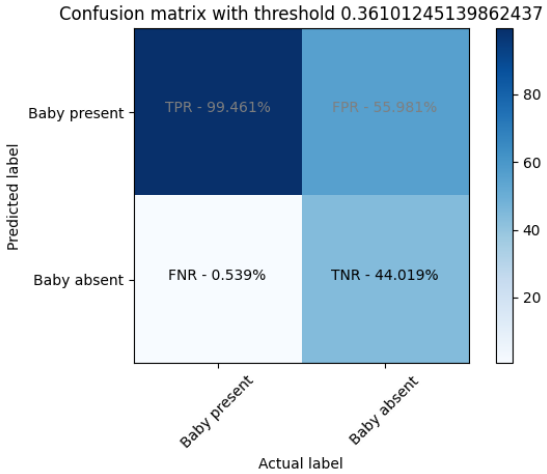

Fig 28: Confusion matrix with threshold of SVM model

V. CONCLUSION

In order to detect the presence of a baby in a carriage on the passenger seat of a car, data was collected using an ultrasonic sensor along with the Red Pitaya STEMLAB measurement board and the Ultrasonic sensor SRF02. Ultrasonic waves interact differently with various surfaces, leading to diverse reflection patterns and frequency profiles. These reflected ultrasonic waves display unique patterns upon being received by the sensor and analysed through the Red Pitaya device.

Three machine learning models i.e., MLP, CNN and SVM algorithms are used to build and train the models. From the results and discussion, we can infer that the MLP model performs better than that of the CNN and SVM models in many parameters while having lower execution time than SVM. Further to improve the accuracy of the model, a threshold feature is introduced for which the model is tuned based on the Confusion matrix parameters such as Accuracy, precision,

The classification models in this project have helped the research in this direction even though we may not fully comprehend the features that are contained in the data. Overall increasing the amount of training data is the only sure-fire approach to raise classification rates. Collective effort can be used by training larger pre-trained models with the data from our experiment to distinguish between baby present in the baby seat or baby absent in the baby seat.

VI. ACKNOWLEDGMENT

We would like to express our gratitude to Prof. Dr. Peter Nauth and Prof. Dr. Andreas Pech for giving us the chance to work on this topic under their supervision and for guiding us in the right direction to finish the required project and report. We also want to express our gratitude for Prof. Julian Umansky important contribution in providing all the tools necessary for the task to be completed successfully. The course "Autonomous Intelligence System" and "Machine Learning" were very beneficial in providing the background knowledge and motivation for learning more on machine learning topics and sparking interest in this field.

REFERENCES

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

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

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

[4] O. Kembuan, G. Caren Rorimpandey and S. Milian Tompunu Tengker, "Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow," 2nd International Conference on Cybernetics and Intelligent System (ICORIS), 2020. [Online]. Available: doi: 10.1109/ICORIS50180.2020.9320810..

[5] Hayder Hasan, Helmi Z.M. Shafri and Mohammed Habshi, "A Comparison Between Support Vector Machine (SVM) and Convolutional Neural Network (CNN) Models For Hyperspectral Image Classification," iopscience, 2019. [Online]. Available: DOI: 10.1088/1755-1315/357/1/012035.

[6] E. N. Budisusila, M. Khosyi'in, S. A. D. Prasetyowati, B. Y. Suprapto and Z. Nawawi, "Ultrasonic Multi-Sensor Detection Patterns On Autonomous Vehicles Using Data Stream Method," 2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), [Online]. Available: doi: 10.23919/EECSI53397.2021.9624313..

[7] K. Balani, S. Deshpande, R. Nair and V. Rane, "Human detection for autonomous vehicles," IEEE International Transportation Electrification Conference (ITEC), 2015. [Online]. Available: doi: 10.1109/ITEC-India.2015.7386891..

[8] Guo, P.; Shi, H.; Wang, S.; Tang, L.; Wang, Z., "An ROS Architecture for Autonomous Mobile Robots with UCAR Platforms in Smart Restaurants. Machines," Machines, 2022. [Online]. Available: https://doi.org/10.3390/machines10100844.

[9] A. H. Pech, P. M. Nauth and R. Michalik, "A new Approach for Pedestrian Detection in Vehicles by Ultrasonic Signal Analysis," IEEE EUROCON 2019 - 18th International Conference on Smart Technologies, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8861933.