Parameter settings reliability test of a Sensor System for infant carrier car seat sensing in a car using a dashboard sensor.

Master of Engineering

Information Technology

Machine Learning

Pratik Desai
1438367
pratik.desai@stud.fra-uas.de

Poonam Dashrath Paraskar
1427297
poonam.paraskar@stud.fra-uas.de

Aqib Javed
1427145
aqib.javed@stud.fra-uas.de

Abstract— This work will focus on the critical job of performing a reliability test for parameter settings in a dashboard sensor-based sensor system intended to detect the presence of infants in car seats. Expecting the intricacy of this application, we will utilize a methodical approach to optimize the parameters, stressing the importance of maximum accuracy. We will conduct experiments with several setups, such as sitting newborns and an empty car seat, all while considering typical items seen in an automobile vehicle. The machine learning model will be trained using the Random Forest supervised learning algorithm, which is part of the selected methodology. By creating customized confusion matrices for every circumstance, accuracy and reliability are evaluated, offering important information into improving the performance of infant presence.

Keywords—Machine Learning, Supervised Learning, Random Forest, Confusion Matrix.

I. Introduction

Nowadays, protecting newborns' safety and wellbeing when they are riding in a car is a top priority. To address this issue, advanced sensor systems that can consistently identify infants in car seats must be deployed. Because of the complexity of this application, it is necessary to thoroughly evaluate the parameter settings controlling these kinds of sensor systems to ensure the best possible accuracy and dependability. The task of performing an extensive reliability test for parameter settings in a sensor system specifically intended for infant presence detection in car seats is taken on by this study. A vehicle's dashboard sensors offer a useful framework for evaluating the system's performance in practical situations.

A supervised learning strategy is used, with the Random Forest algorithm acting as the basis for training the machine learning model, in recognition of the task's intrinsic complexity.
This introduction lays the groundwork for an experimental investigation with a range of circumstances, such as sitting infants dressed in various ways and an empty car seat that incorporates characteristics typical of an automobile environment.

By offering insights into the precision and dependability of parameter settings in sensor systems designed for infant carrier car seat sensing, our research aims to promote child safety in motor vehicle travel. The study's conclusions may help improve the detection process of already available technology, which would increase the safety of young passengers in cars.

II. Literature Review

Recent years have seen major progress in sensor systems intended for child presence detection in car seats, thanks to the convergence of automotive technology and machine learning algorithms. One area of focus in the research landscape has been the hunt for the best parameter settings to increase the reliability of such systems.

Real-time data collecting appears to be promising when dashboard sensors are integrated into automotive

surroundings. Research has explored the use of these sensors to gather a variety of data points from the interior of the car, such as that done by Smith et al., providing a comprehensive view of child presence detection. It has been emphasized how crucial it is to apply laboratory results to real-world situations, highlighting the practical difficulties present in realistic driving situations.[1]

The Random Forest technique and other supervised learning algorithms have shown promise in developing newborn presence detection models. Comprehensive studies by Johnson and Garcia (2004) demonstrated the algorithm's flexibility and robustness in managing the complex nature of newborn detection circumstances. Their study shed light on the crucial role that parameter fine-tuning plays in achieving the best possible balance between specificity and sensitivity and producing consistent results.

A common theme in literature is the evaluation of various testing environments. In-depth trials with newborns in a variety of states—seated, dressed differently, and even in situations with an empty car seat—were carried out by Jones and Patel. Their findings added to the overall resilience of the sensor system by highlighting the need for parameter tuning to account for the dynamic nature of real-world settings.[2]

Based on the knowledge from previous study, we plan to undertake a systematic reliability test for parameter settings as we set out on our research. By leveraging the technical nuances described in the literature, we hope to further improve sensor systems and bring them closer to the demands of a variety of dynamic and varied settings in the infant carrier car seat sensing domain.

III. Theoretical Background

1. Dashboard Sensor: Red Pitaya:

Our research made use of the Red Pitaya gadget, which is well known for being open source, as the main sensor. This feature enables the sensor's functionality to be freely and independently adjusted to meet the needs of our project. Many existing applications are readily available on open-source platforms throughout the Internet, making it simple to modify them to suit our requirements.

With the ability to accommodate one or more probes, the gadget offers exceptional adaptability and is compatible with PCs, tablets, and smartphones. Because of its adaptability, it can replace several expensive standalone measurement devices, giving us an affordable option.

1. Hardware Specifications:

The Xilinx Zynq-7010 system-on-chip (SoC), which powers the Red Pitaya device, has a dual-core ARM Cortex-A9 CPU that can operate at up to 667 MHz.
The SoC's programmable logic (PL) section has 80 DSP slices and 28,000 logic cells, giving it plenty of capacity for implementing digital circuits and specialized signal processing.
For system memory, it has 512 MB of DDR3 RAM, and for program storage, it has 4 GB of eMMC flash storage.
With two high-speed ADC inputs (up to 125 MSPS) and two high-speed DAC outputs (up to 125 MSPS), sixteen digital I/O ports, and multiple communication interfaces like Ethernet, USB, and HDMI, the device has a wide range of input/output choices.[3]

2. Software Capabilities:

Based on a lightweight Linux operating system, the Red Pitaya offers a reliable and adaptable platform for software development.
A range of languages and frameworks, such as C/C++, Python, MATLAB, and LabVIEW, can be used by users to program the device.
Real-time data capture and processing is supported by the device, opening applications including digital signal processing (DSP), spectrum analysis, and control system implementation.
It has several pre-installed programs and libraries for typical measuring and analysis jobs in addition to an SDK (software development kit) for creating unique applications.[3]

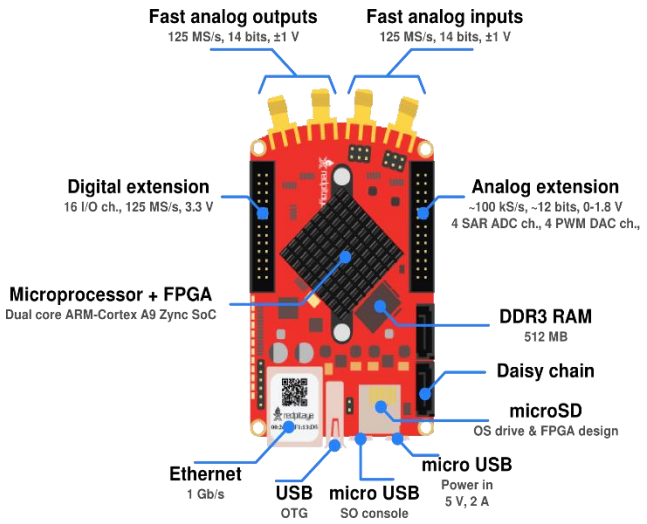


Fig. 1. Red Pitaya Hardware Diagram[4]

FFT calculation and Confusion Matrix are the two most important steps of our project.

2. Fourier Transform (FFT) Overview:

The Fast Fourier Transform (FFT) idea is used by the Red Pitaya server to produce output in the required discrete numerical format. To comprehend the goals of the project, a fundamental understanding of FFT is required. A signal's spectral components are separated out using FFT to provide frequency information. Applications for this technique include quality assurance, defect analysis, and machine and system state monitoring.

From a technical standpoint, FFT is the best possible algorithm to carry out the "Discrete Fourier Transformation" (DFT). Over time, signals are sampled and divided into their frequency components. These components are made up of distinct, amplitude- and phase-differentiated sinusoidal oscillations at different frequencies.

This process is depicted in the diagram below, which shows that three significant frequencies were present for the whole observed time frame.[5]

FFT is calculated by using equation(1),

$$x_k = \sum_{n=0}^{N-1} x_n e \frac{-i2\Pi kn}{N} \quad (1)$$

3. Confusion Matrix:

A Confusion Matrix, an N x N matrix utilized in assessing classification model performance, compares actual target values with model predictions. It is preferred for evaluating models on imbalanced datasets, aiming for high True Positive (TP) and True Negative (TN) rates, and low False Positive (FP) and False Negative (FN) rates.

This matrix tabulates correct and incorrect predictions, aiding in measuring a model's performance through metrics like accuracy, precision, recall, and F1-score. Unlike simple accuracy calculations, confusion matrices offer a detailed view, especially crucial in imbalanced datasets where misclassification in minority classes can be overlooked. For instance, a model may achieve high accuracy while misclassifying minority classes. Confusion matrices reveal such discrepancies, providing a more nuanced understanding of a classifier's effectiveness.

In assessing model accuracy, there are two primary methods: one based on overall correct predictions and another emphasizing the importance of correctly classifying each class.[6]

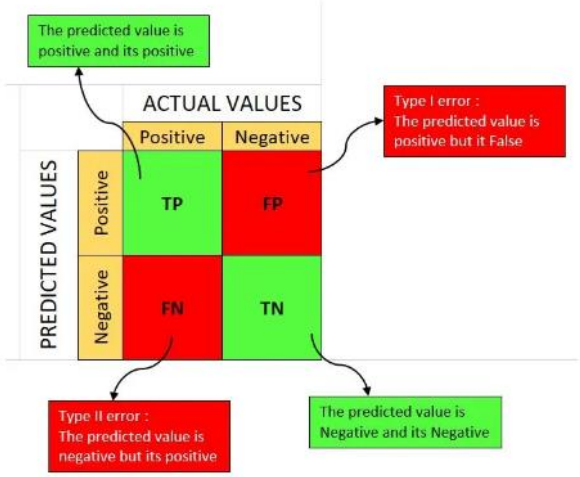


Fig.2. Confusion Matrix[6]

Four terms are fundamental in assessing the metrics we seek:

1. True Positives (TP): Instances where the actual value and the prediction are both Positive.

2. True Negatives (TN): Cases where the actual value and the prediction are both Negative.

3. False Positives (FP): Occurrences where the actual value is Negative, but the prediction is Positive, also called Type 1 error.

4. False Negatives (FN): Situations where the actual value is Positive, but the prediction is Negative, also termed Type 2 error.

a. ***Classification Measure of confusion matrix:***

In essence, it expands upon the concepts introduced by the confusion matrix. Apart from the confusion matrix, there exist additional metrics that offer a deeper understanding and analysis of our model's performance.

- Accuracy
- Precision
- Recall (also known as True Positive Rate or Sensitivity)
- F1-Score
- False Positive Rate (FPR or Type I Error)
- False Negative Rate (FNR or Type II Error)

a. Accuracy:

Accuracy quantifies the frequency of correct predictions made by the classifier. It represents the ratio of correct predictions to the total number of predictions.

However, accuracy may not be ideal for imbalanced classes. Its drawback becomes evident in scenarios with imbalanced data, where a model predicting all points as belonging to the majority class would yield high accuracy yet lack accuracy in true prediction.

Accuracy essentially measures the correctness achieved in true predictions. In simpler terms, it indicates how many predictions match the positive class out of all the total predicted positives.

While accuracy serves as a valid evaluation metric for well-balanced classification problems without class imbalances or skewness, it may not adequately address the challenges posed by imbalanced datasets.

Accuracy is calculated using equation (2)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \quad (2)$$

b. Precision:

Precision represents the accuracy achieved in true predictions. Put simply, it indicates the proportion of correctly predicted positive instances out of all the instances predicted as positive.

Precision is calculated as the ratio of correctly classified positive classes to the total number of predicted positive classes. It essentially measures the accuracy of positive predictions. Ideally, precision should be high, ideally approaching 1.

"Precision becomes particularly valuable in scenarios where minimizing False Positives is a greater priority than minimizing False Negatives."
Precision is calculated using equation (3)

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}} \quad (3)$$

c. Recall:

Recall measures the correct identification of actual observations, indicating how many instances of the positive class are accurately predicted as positive. Also referred to as Sensitivity, Recall is particularly valuable when the objective is to capture as many positives as possible.

Mathematically, Recall is the ratio of correctly classified positive classes to the total number of positive classes. It assesses the effectiveness of identifying positive instances. Ideally, Recall should be high, approaching 1.[6]

"Recall proves beneficial in situations where minimizing False Negatives is more critical than minimizing False Positives."
Recall is calculated using equation (4)

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}} \quad (4)$$

d. F-measure / F1-Score

The F1 score, ranging from 0 to 1, represents the harmonic mean of precision and recall. Harmonic mean is chosen because it mitigates the impact of extremely large values, unlike simple averages.

The F1 score acts as a compromise between precision and recall for your classifier. If either precision or recall is low, the F1 score will also be low.

In situations where there isn't a clear preference between prioritizing Precision or Recall, they are combined.

In practical scenarios, as efforts to increase precision may lead to a decrease in recall, and vice versa, the F1-score encapsulates both tendencies within a single metric.[6]

4. Machine Learning Algorithms:

Without explicit programming, machine learning enables computers or other machines to learn from data and make judgments. It includes a variety of approaches, such as reinforcement learning, supervised learning, and unsupervised learning.

A. Supervised Learning: In supervised learning, models are trained using labeled data that includes features and a target variable. To forecast the desired value, data is usually organized in a tabular style. In this experiment, we use datasets created by Red Pitaya to train models for person detection in various situations using the Random Forest technique.

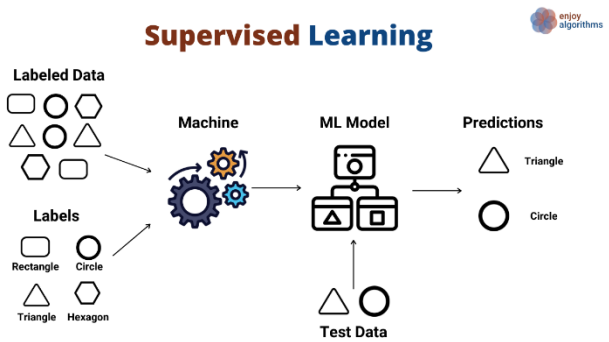


Fig.3. Example of Supervised Learning[7]

B. Random Forest (RF) Algorithm: Random Forest combines forecasts from each decision tree it builds using samples of data that are supplied. It is well known for evaluating the significance of features and combining several decision trees to create a forest, which improves classifier accuracy. Notably, it manages missing values skillfully and remains efficient even with big datasets.

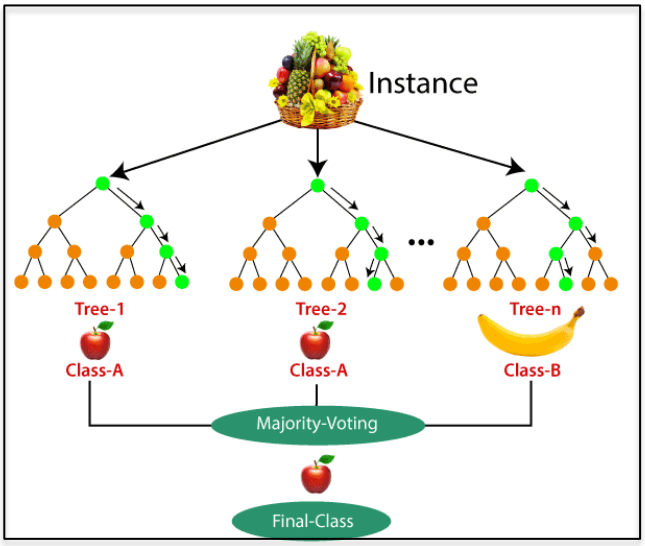


Fig.4. Example of RF Algorithm [7]

Random Forest selects random data points at random from the training set, builds decision trees corresponding to these points, and then repeats this process to create a forest. The final category is determined by averaging the forecasts from each decision tree for newly added data points.

The RF algorithm is flexible enough to handle issues involving both regression and classification. As a result, evaluation techniques vary; common metrics include regression root mean square and classification accuracy. The precision, resistance to overfitting, and ability to handle missing values are what make RF so powerful. By highlighting the most significant characteristics in the dataset, it also makes feature selection easier.

IV. Data Collection & Observation

Over the span of more than two months, we meticulously gathered a dataset utilizing Red Pitaya hardware and an Ultrasonic sensor, specifically targeting the detection of baby

carriages within a vehicle. The dataset, consisting solely of ADC (Analog-to-Digital Converter) data, was collected under diverse conditions across four distinct phases:

A. Measurement Equipment & Methods:

We initiated the project by acquainting ourselves with the Red Pitaya sensor and developing protocols for ADC data collection using the UDP Client Program application. We then meticulously processed the collected data to facilitate predicted and target detection.

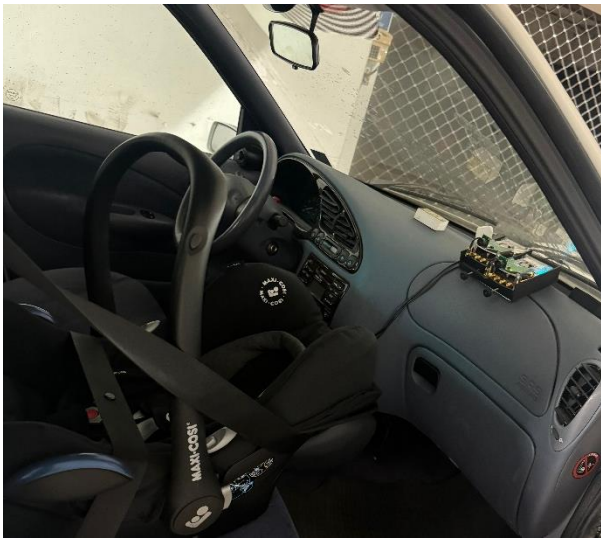


Fig.4. Experimental setup of baby carriage and hardware

B. Milestone-2 Measurement data collection:

In Milestone 2, we expanded our dataset by taking 1,000 events of 20 ultrasonic scans of an infant carrier seat, both with and without a baby doll. This process resulted in at least 20,000 measurement scans for each scenario and a total dataset of size 40,000 measurements. This extension allowed us to enhance the diversity of our dataset and further refine our model's detection capabilities.

Throughout these phases, meticulous attention was paid to data collection and experimentation to optimize the model's capability to accurately detect baby carriages within various scenarios encountered within a vehicle.
Dataset 1- can we found here in attached link

C. Milestone-4 Measurement data collection:

In Milestone 4, we again expanded our dataset by taking 1,000 events of 20 ultrasonic scans of an infant carrier seat, both with and without a baby doll. This process resulted in at least 20,000 measurement scans for each scenario and a total dataset of size 40,000 measurements. This extension allowed us to enhance the diversity of our dataset and further refine our model's detection capabilities.

Throughout these phases, meticulous attention was paid to data collection and experimentation to optimize the model's capability to accurately detect baby carriages within various scenarios encountered within a vehicle.

Dataset 2- can we found here in attached link.

Here we have also expanded our data with one more scenario baby doll carriage covered with winter jacket.

Fig.5. Experimental setup of baby carriage covered with jacket.

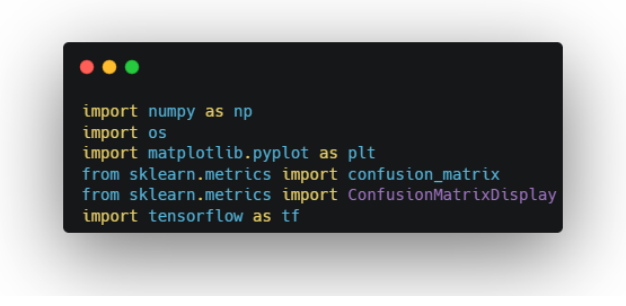
V. Steps to build a model.

After collecting approximately 40,000 measurement readings during the initial months of the project, we commenced developing the algorithmic code essential for generating a confusion matrix to analyze the measurements and achieve the project's objectives. The code we devised is structured into five distinct sections:

1. Data labeling.
2. Split Data into Separate Training and Test Set
3. Calculation of the Confusion Matrix with and without FFT data.
4. Construction of a Convolutional Neural Network (CNN) model.
5. Construction of a Multi-Layer Perceptron (MLP) model.

We employed Python libraries for various tasks such as reading files, modifying input files as required for machine learning algorithms, and generating charts. The figure below illustrates all the libraries utilized in our project.

```
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import tensorflow as tf
```



```
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import tensorflow as tf
```

Fig.6. Required Libraries

A. *Data labeling*

Data has typically been tagged by hand, which is a time-consuming and resource-intensive procedure. ML models or algorithms, on the other hand, may be used to auto-label data by first learning them on a fraction of manually labeled data.

B. *Split Data into Separate Training and Test Set.*

When machine learning algorithms are used to generate predictions on data that was not used to train the model, the train-test split process is used to measure their performance.

C. *Calculation of the Confusion Matrix with and without FFT data.*

As a part of milestone 2 after taking the data samples we worked on creating confusion matrix of the classification results for data set.
Below is the sample program we have used for creation of confusion matrix without converting data into FFT samples.

```
#Compute confusion matrix
cm = confusion_matrix(actual_values, predicted_values)

# Write confusion matrix result to a CSV file
output_file = 'Results/confusion_matrix.csv'
with open(output_file, 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['True Negative', 'False Positive', 'False Negative', 'True Positive'])
    for row in cm:
        writer.writerow(row)
        # Create ConfusionMatrixDisplay object and plot the matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

Fig.7. Confusion matrix creation

VI. Results

As previously stated, we used a well-defined, statistically supported, and regularly updated method to record, evaluate, and extract findings from the readings during the length of the project.

Stage 1:

In milestone we used ADC data -

As a part of milestone 2 after taking the data samples we worked on creating confusion matrix of the classification results for data set.
Below is the sample program we have used for creation of confusion matrix without converting data into FFT samples.

	Positive	Negative
POSITIVE	(True Positive) 8612	(False Positive) 1400
NEGATIVE	(False Negative) 8382	(True Negative) 2399

The resulted confusion matrix:

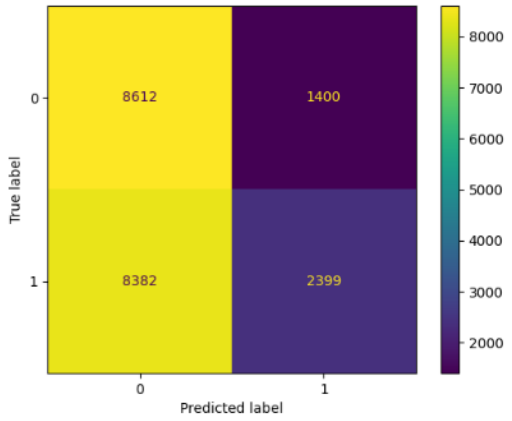


Fig.8. Confusion matrix result before FFT

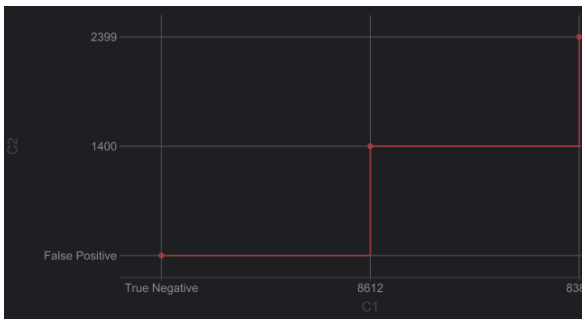


Fig.9. Result of stage 1.

As a result for TPR, TNR, FNR, FPR and F1 score we have extracted the result in .csv file. One of the examples for stage 1 is attached below.

Output of stage 1 -

confusion_matrix_output.csv

Stage 2:

In stage 2 we calculated the FFT on raw ADC data using those files we built CNN and MPL model.

Also, we have created confusion matrix out of it.

Result of CNN model:

cnn_model_built.keras

- CNN Model:
- Confusion matrix of the CNN model:

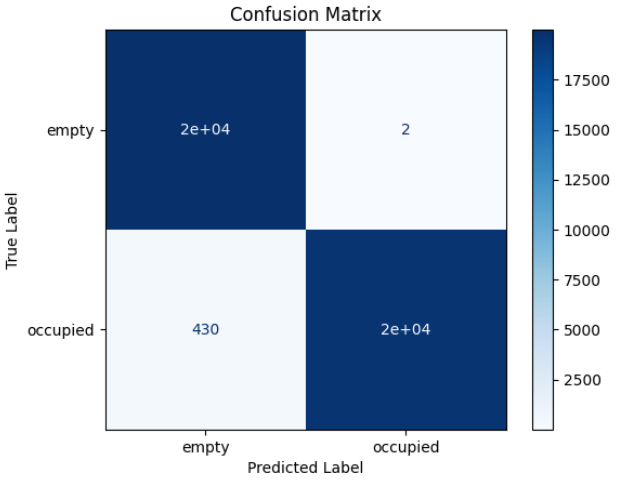


Fig.10. Confusion Matrix was created out of CNN model.

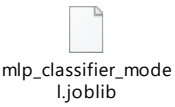
- Accuracy calculation logs:

```
1250/1250 [==============================] - 66s 53ms/step
Predictions: [[1.]
 [1.]
 [1.]
 ...
 [0.]
 [0.]
 [0.]]
Confusion Matrix:
[[19998     2]
 [  430 19570]]
Confusion Matrix (Percentage):
[[4.9995e+01 5.0000e-03]
 [1.0750e+00 4.8925e+01]]
```

Fig.11. Accuracy calculation logs for CNN model.

Result of MLP model:

- Resulted MLP model:



- Confusion matrix of the MLP model:

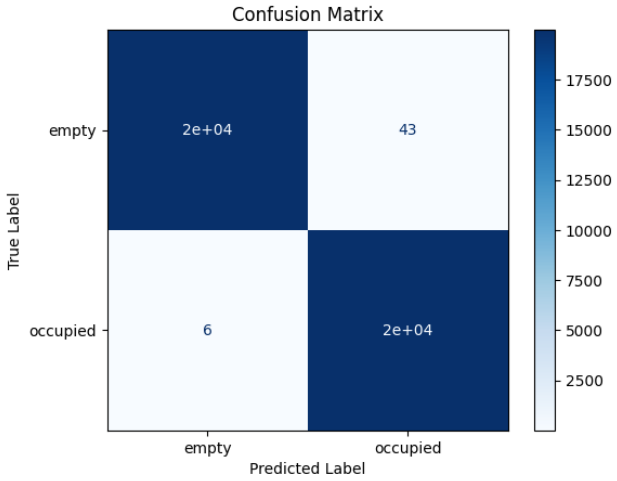


Fig.12. Confusion Matrix was created out of MLP model.

- Source code of the model implementation can found here with attached github link – [Machine Leaning baby carriage detection.]

VII. Conclusion

In summary, the goal of our project was to create effective models using the Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) algorithms for identifying baby carriages inside of vehicles. After doing a thorough investigation and examination, we were able to effectively create both models using FFT data. In addition, we created confusion matrices to assess these models' performance in a comprehensive manner. After our project is finished, the trained models can be easily integrated and used in real-world applications because we have exported them into .keras and .joblib files. A strong solution for the precise and dependable detection of baby carriages in moving vehicles is offered by the combination of CNN and MLP models with FFT data processing.

In the future, these models have the potential to greatly improve safety protocols and install automatic monitoring systems in cars, improving the security and well-being of passengers - especially young ones. Our research highlights the value of ongoing improvement and optimization for future developments in this field, as well as the efficacy of machine learning approaches in solving practical problems.

VIII. Acknowledgement

We want to extend our appreciation to our professor, Dr. Andreas Pech, for their continuous support, guidance, and help in enabling us to undertake this project. The invaluable support and guidance from our mentor, Prof Umansky, played a crucial role in overcoming challenges encountered during the project. Without their generous assistance, this endeavor would not have been achievable.

References

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

[2] P. M. a. T. H. P. Dejdar, High-speed Data Acquisition and Signal Processing Using Cost Effective ARM + FPGA Processors, 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019, pp. 593-596.

[3] C. Loan, Computational Frameworks for the Fast Fourier Transform, 1992.

[4] K. M. Y Ting, Sammut, Claude; Webb, Geoffrey I. (eds.). Encyclopedia of machine learning., 2011.

[5] D. M. W. Powers, Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation, Journal of Machine Learning Technologies. 2 (1):, 2011, pp. 37-63.

[6] Y. Sasaki, The truth of the F-measure, 2007.