Milestone 3 – fft dataset#1 , trained MLP and Confusion matrix.

(Parameter setting and reliability test of a sensor system for infant carrier car seat sensing in a car using a dashboard sensor)

Masters of Engineering
Information Technology

Aneeta Antony(1431461)
aneeta.antony@stud.fra-uas.de

Sonam Priya (1427129)
sonam.priya@stud.fra-uas.de

Nitu Shrestha (1428296)
nitu.shrestha@stud.fra-uas.de

I. IMPLEMENTATION

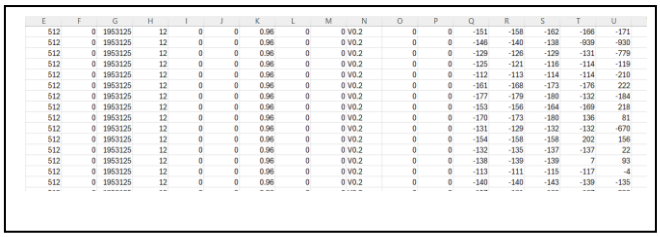
A. Calculate fft of all adc measurements.

With The Red-pitaya sensor we measured and saved the dataset 1. Firstly, data is collected with baby doll placed in a carrier in the passenger seat, secondly another set of reading without baby doll is also noted. The scanned data from sensor is then stored with ADC data and header details.

This data is now transformed to fft data using appropriate python libraries and coding techniques.

1) Preprocessing the ADC Dataset

The ADC dataset with and without baby doll is properly labeled and saved in separate text files. This is converted as csv files for the convenient use in our programming. Figure 1 shows the collected ADC data as a csv.



E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-151	-158	-162	-166	-171
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-148	-140	-138	-939	-930
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-129	-126	-129	-131	-779
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-125	-121	-116	-114	-119
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-112	-113	-114	-114	-210
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-161	-168	-173	-176	222
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-177	-179	-180	-132	-184
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-153	-156	-164	-169	218
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-170	-173	-180	136	81
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-131	-129	-132	-132	-670
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-154	-158	-158	202	156
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-132	-135	-137	-137	22
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-138	-139	-139	7	93
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-113	-111	-115	-117	-4
512	0	1953125	12	0	0	0.96	0	0	V0.2	0	0	-140	-140	-143	-139	-135

Figure 1

2) Convert data set to fft

The data set#1 is converted to fft using NumPy inbuilt functions. And this is saved as NumPy data frame array for further use like training the MLP etc. During the conversion we appropriately label the data as 1 or 0, for babydoll presence detected and no babydoll respectively.

```
# Selecting data from the 17th column to the end of the DataFrame
adc_data_selected_columns = combined_df.iloc[:, 16:].mean(axis=1)
# Converting the selected pandas Series to a numpy array
adc_array = adc_data_selected_columns.to_numpy()
# Choosing a window function for signal processing, here using the Hanning window
window = np.hanning(len(adc_array))
# Applying the chosen window function to the data
windowed_adc_data = adc_array * window
# Performing Fast Fourier Transform (FFT) on the windowed data
fft_result = np.fft.fft(windowed_adc_data)
```

Figure 2

B. Creation of an MLP.

1) Choise of Machine learning model :- Random Forest Classification Modeling

Here in our project, we decided to try with a Random Forest (RF) classifier for modeling. Random Forest is a classifier comprising multiple decision trees trained on different subsets of the dataset. It then aggregates their predictions to enhance the overall predictive accuracy. Instead of relying on a single decision tree, Random Forest leverages the collective predictions of multiple trees, determining the final output based on the majority vote. The inclusion of a greater number of trees in the forest improves accuracy and helps mitigate overfitting [1].

The RF classifier was developed in Python utilizing the Scikit Learn machine learning framework. Scikit Learn seamlessly integrates with Python's NumPy and SciPy libraries, offering various classification algorithms, including Random Forest.

The dataset is divided into two types of data arrays: training and testing. Manually partitioning the dataset is unnecessary, as Scikit-learn's train-test split function automatically divides the dataset into random subsets. Additionally, a random state is specified for ensuring reproducibility of the operation.

The random forest (RF) is a tree-structured ensemble learning method. The basic idea of building a random forest is [2]: First, randomly extract *K* new sample sets from the original data set with bootstrap resampling method, and use the new samples to construct *K* classification decision trees; then, suppose there are *m* feature, randomly select m_{try} features as candidate split attributes, and select one of m_{try} feature attributes for splitting according to the Gini index; the above process would be iterative executed, and finally, without cutting the decision tree, directly integrate the generated decision trees form a random forest to classify the new data, and the classification results are obtained by the majority voting strategy, the class with the most votes is the final classification result.

RF constructs different training sets to increase the difference between classification models, thereby improving the extrapolation prediction ability of combined classification models. Through k rounds of training, a classification model sequence $\{h_1(X), h_2(X), \ldots, h_K\{X)$ is obtained, and then they are used to form a multi-classification model system. The final classification result of the system adopts a simple majority voting method. Final classification decision is :

.

$$H(x) = \{arg_Y^{max}\} \sum_{\{i=1\}}^{K} I\left(\{\{h_i\}(x) = Y\}\right)$$

Where $H(x)$ is a combined classification model, h_t is a classification model of a single decision tree, Y is an output class, and I(·) is an indicative function. The above equation illustrates the use of majority voting to determine the final classification[3].

2) Create and train MLP using the dataset#1

The Total dataset is randomly split as test and train data and used in RF Classifier training.

```
# Splitting the dataset into the training set and
test set
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.3,
random_state=42
```

```
# Initialize RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
# Get the best parameters and best estimator
best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_
# Evaluate the best model on the test set
y_pred = best_estimator.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Figure 3 RF Classifier

3) The hyper parameter tuning used for the RF Classifier for improvemts.

```
param_grid = {  'n_estimators': [50, 100, 150],
# Number of trees in the forest
   'max_depth': [None, 10, 20],  # Maximum
depth of the tree
   'min_samples_split': [2, 5, 10],  # Minimum
number of samples required to split an internal
node
 'min_samples_leaf': [1, 2, 4]  # Minimum
number of samples required to be at a leaf node
}
```

C. Confusion Matrix created from trained MLP

Toward this milestone we generated a confusion metrix from the MLP created. Comparing the predicted labels with test labels. Predicted label and accuracy are found for the dataset#1 test data.

1) The Confusion Matrix is obtained form the trained MLP(RF Classiffer model) in Figure 4 show between "With Baby" and "Without Baby."

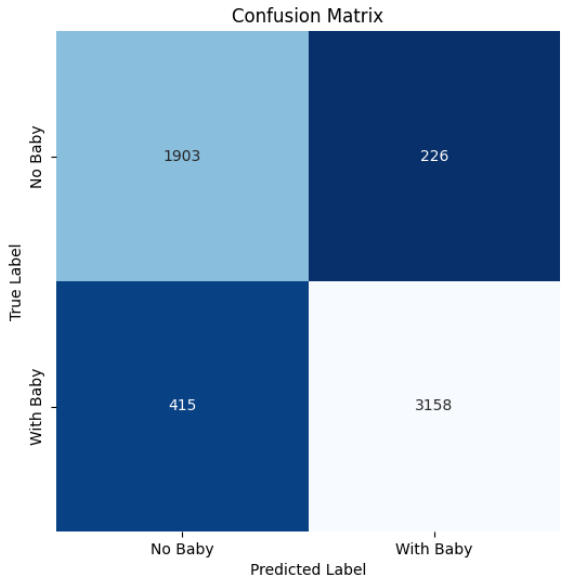


Figure 4 Confusion Matrix

2) The Accuracy and Precision details of the current model

- Accuracy: 0.89
- Precision: 0.93

- Recall: 0.88
- F1-score: 0.91
- Best parameters found by grid search:
- {'max_depth': None,
  'min_samples_leaf': 4,
  'min_samples_split': 2,
  'n_estimators': 150}

- Best accuracy on validation set: 0.89
- Accuracy on test set: 0.89

3) Save trained MLP for further use.

The RF Classifier model is after training saved for further use on the dataset3.

```
# Save the best_estimator to a file
joblib.dump(best_estimator,random_forest_model.pkl')
```

D. FFT Dataset.

The transformed fft dataset#1 can be found in the OneDrive Link - FFT-Dataset1 (processed_data folder). The fft data is processed and saved as numpy dataframe arrays are saved for the with-baby-doll and with-out-baby-doll namely withbaby_npy_array & withoutbaby_npy_array.

REFERENCES

[1] K. S. Gill, V. Anand, R. Gupta and P. -A. Hsiung, "Detection of Malware Using Machine Learning techniques on Random Forest, Decision Tree, KNeighbour, AdaBoost, SGD, ExtraTrees and GaussianNB Classifier," 2023 4th IEEE Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2023, pp. 1-7, doi: 10.1109/GCAT59970.2023.10353495. keywords: {Codes;Digital systems;Computational modeling;Malware;Stability analysis;Data models;Pattern recognition;Data Science;Behaviour Analysis;Malware Attacks;Anomaly Detection;Classifcation;Machine Learning;Deep Learning;Artifcial Intelligence;Classifers}

[2] K.N. Fang, J.B. Wu, J.P. Zhu and B.C. Xie, "A review of technologies on random forests", *Statistics and Information Forum*, vol. A26, pp. 32-38, 2011.

[3] D. Yuan, J. Huang, X. Yang and J. Cui, "Improved random forest classification approach based on hybrid clustering selection," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 1559-1563, doi: 10.1109/CAC51589.2020.9326711. keywords: {Random forests;Decision trees;Clustering algorithms;Classification algorithms;Indexes;Vegetation;Partitioning algorithms;random forest algorithm;clustering ensemble selection;personal indoor thermal preference}

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html