Investigating ultrasonic signals in discriminating different objects

M.Eng. Information Technology
Individual Project
Prof. Dr. Peter Nauth

HOANG LONG, NGUYEN
1067534
hoang.nguyen3@stud.fra-uas.de

***Abstract*—Within the scope of this document, I have introduced an approach to discriminate when human occupied an empty seat or not, in the scenarios within an automobile space. By analyzing the time-frequency dependency of the echo signals, perceiving by the REDPITAYA sensor, I have used Gabor transformation to extract the high-level features, that appear in the spectrograms and can be exploited to discriminate the signal sources. To classify the ultrasonic feedbacks, I applied the well-known machine learning model: Convolutional Neural Network (CNN) for its high performance in image-like data. As the result, the workflow I have designed have achieved a significant reliability in detecting two cases: "human" or "empty" within MATLAB environment.**

Keywords—ultrasonic echo signal, human and empty space, Gabor transformation, spectrogram, machine learning, Convolutional Neural Network, REDPITAYA sensor, MATLAB.

DISCLAIMER

I strongly confirm that this report is a product of my personal work. No other sources were used except those referenced.

The content which is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by me or are provided with an appropriate reference. This work has not been submitted in the same or similar form or to any other examination board.

I also declare that all opinions, results, conclusions, and recommendations are my own and may not represent the policies or opinions of Frankfurt University of Applied Sciences (FRA-UAS).

I. INTRODUCTION

This project is carried out to qualify the requirements of the course module: Individual Project under the supervision of Prof. Dr. Peter Nauth, belonging to the program M.Eng. Information Technology at FRA-UAS.

This report presents the description of project, the test data acquisition and dataset construction, briefly literature review about the techniques in Gabor transform and CNN, the discussion on the result of the project: "Investigating ultra-sonic signal in discriminating different objects".

During the project, the hardware tool I have used is Devantech SRF02 ultrasonic sensor located on the REDPITAYA board for echo signal acquisition. The software are MATLAB for data analysis and spectrogram production, dataset building, CNN model training and testing. I also use MATLAB app designer for making of an extra Graphical User Interface (GUI) for future work.

II. LITERATURE REVIEW

A. Gabor Transformation

Gabor transformation is named after Dennis Gabor, which describing a short-time Fourier transformation. In time-frequency analysis, Gabor transformation is used to determine the sinusoidal frequency and phase content of a small area of the signal as it variates overtime. Firstly, the signal function that we are aimed to analyze, is multiplied by a Gaussian function, which acts as a window function. Then the window will be sided over the signal as the time changes to calculate the output of the transformation. The mathematical formula of the transformation can be deduced as following : [1]

$$G_x(\tau,\omega) = \int_{-\infty}^{\infty} x(t) e^{-\pi(t-\tau)^2} e^{-j\omega t}\, dt$$

At the end of the calulations, every results of Fourier transformation will be stacked up to contruct a new image-like data, which is called as spectrogram. It contains the information of how the signal changes gradually and how much the dominant frequencies it has. With this essential information, I can apply different methods to classify the signal sources.

B. Convolutional Neural Network (CNN)

In machine learning methods, a neural network is way to mirror the function of the human brain. It is constructed by multiple layers, which can be defined as input, hidden and output. There are nodes in the layer of the network, which is mimic the behavior of our brain neurons. The nodes is connected to the adjacent layers and their values are assigned to each connection called as weight values. To calculate the node, a function is used where the input values are multiplied with the corresponding connection values and then put into an activation function. This is a basic understanding about a neural network.

Taking the general concept of neural network, CNN is moving it to a higher level by adding many new parts, such as convolution, sub-sampling, and pooling layers. With these convolution layers, CNN can extract features and construct the feature map. As complex the architecture of the CNN, as complex the feature when multiple layers are stacked. Therefore, these sub-sampling and pooling layer are put in between them.

CNN is strong in detecting high-level features appearing in the input data but keeps reducing memory and computational power during training. That is the reason I

choose one simple architecture of CNN to classify the spectrograms in my individual project. The architecture will be discussed later in the later parts of this report.

III. RELATED WORKS

This individual project used many assets of my previous projects in the same study, especially in the topic :" Audio Classification: CNN Network vs. LSTM Network – A Comparison", "Discrimination of reflected sound signals by applying Gabor transform and CNN classifier"[2],[3]. With the data acquisition part to construct the dataset for the CNN, the project is assisted by Mr. Hoang Hai Nguyen, and his teammates with their project in investigating and gathering sensor feedbacks within an automobile.

IV. MY APPROACH IN DETAIL

A. *The overall workflow of the project*

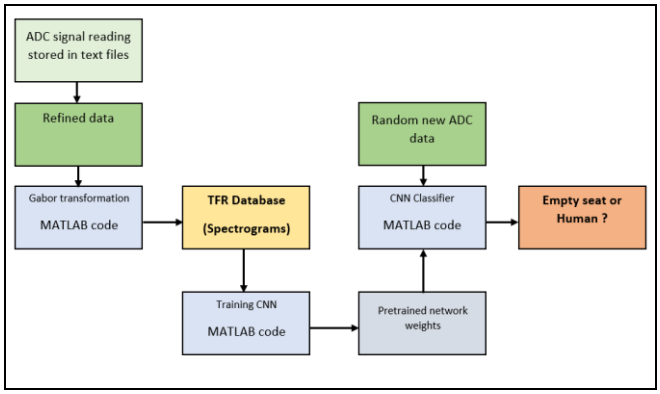


Figure 1: The workflow of my individual project

To tackle to the challenge to discriminate either the seat in the automobile has been taken or not, I have designed a workflow as can be seen in the following Figure 1. In general, I have worked with multiple code files in MATLAB. For some parts, I have utilized some supported toolboxes (libraries) by MATLAB, such as Digital Signal Processing (DSP), Parallel Computing and Machine Learning, thus I do not have to write code for Gabor transform or CNN from scratch.

Firstly, I acquired the ADC signal reading stored as an output text file (.txt) from the REDPITAYA sensor board. This file also contains the prediction of the current fully connected network and some irrelevant information as the first 16 header columns; thus, they are needed to be removed. The raw ADC readings are also ununiform and unbalanced regards hardware noises, that need to be normalized and centered. After the pre-processing steps, the Gabor transform now can be applied to produce the Time-Frequency-Relation or the spectrogram. The time and amplitude relation (2D) will become a time, frequency, and amplitude relation (3D). The spectrogram is an image-like data, that can be treated well with the CNN, where it looks for the high-level features to classify.

After gathering all the spectrograms (images) from two classes, I set up a CNN model to train with them. The model architecture I have used is simple and belongs to MATLAB sample of classify MINIST dataset.[4] The architecture can be tuned as the further improve for this project in the future if the number of classes increases. The weight storing all the information that CNN have learned will be saved and be used again to predict the new ADC input signal. For this part, I have worked with MATLAB APP Designer to create a friendly graphical user interface (GUI).

B. *Data acquisition*

At the beginning, with the assistant of Mr. Hoang Hai Pham and his teammates, I collected 1400 sensor readings of two cases: "human" and "empty seat". In the Figure 2, it can be seen the position of the REDPITAYA sensor in the cabin of the automobile from the seat where it needs to perceive changes. The red box I have drawn, shows the estimate range of the ultra-sonic sensor (rule of thumb). When a human enters and occupied the seat, the sensor echo reading will be marked as 'human", otherwise will be as "empty".

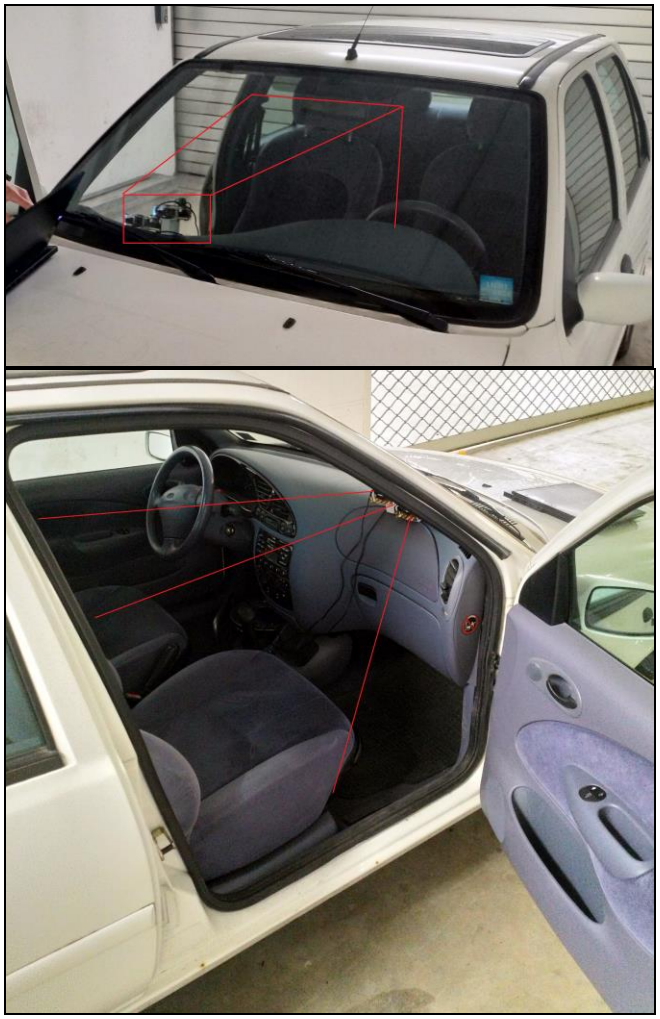


Figure 2: The REDPITAYA sensor position respects to the seat

To increase the diversity of the data collection, there are some additional constraints in two cases, such as either the entrance door is shut or not, the seat belt is put on or not, human moves or stays stable in the cabin. The code names of some outcome possibilities, that are used to train CNN, are listed out in the Table 1. The number coming after the code name indicates the different attempt of collecting the

data, for example, BEC1 means the first time the measure is made, BEC2 means the second one.

Table 1: Code names of gathered data to train the CNN

Code name	Meaning
BEC	seat **B**elt is put on, **E**mpty space with **C**losed door.
BHC	seat **B**elt is put on, **H**uman occupied with strong movement in the cabin space with **C**losed door.
BSHC	seat **B**elt is put on, **H**uman occupied **S**tably the space without any strong movements. The door is **C**losed.
EC	Seat belt is put off, **E**mpty space with **C**losed door.
HC	Seat belt is put off, **H**uman occupied with strong movement in the cabin space. The door is **C**losed.
SHC	Seat **B**elt is put on, **H**uman occupied **S**tably the space without any strong movements. The door is **C**losed.

C. Data pre-processing & Gabor transformation

1) Data pre-processing

As the input signal samples are retrieved from Mr. Hoang Hai Pham and his teammates, it is clear to acknowledge that they are affected by hardware noised and unnormalized. Figure 3 show one of the raw ADC signals deviated from 0 values (denoted as the red horizontal line) and amplitude ranges from -600 to 600. Before applying any frequency analysis techniques, two pre-preposing methods must be done with all the ADC samples, such are centering and normalizing.

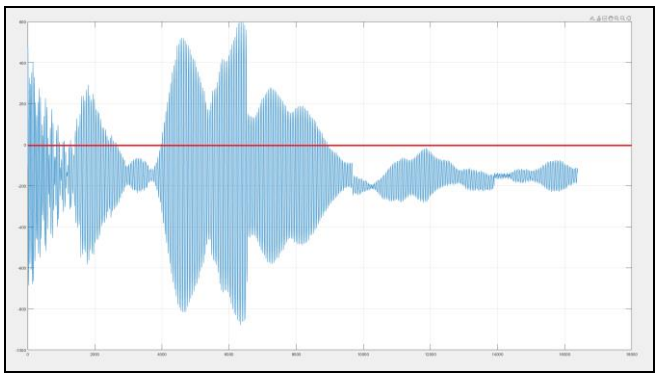


Figure 3: Raw ADC signal (SHC) from REDPITAYA sensor is unnormalized and unbalanced (a deviation from 0, red line)

By using a loop and accessing every 16000 sampling points of each echo sample, it is possible to divide them to their max absolute value to normalize the signal. Additionally, by subtracting each sampling points to their average mean value, the signal will be centralized as illustrated in the Figure 4.

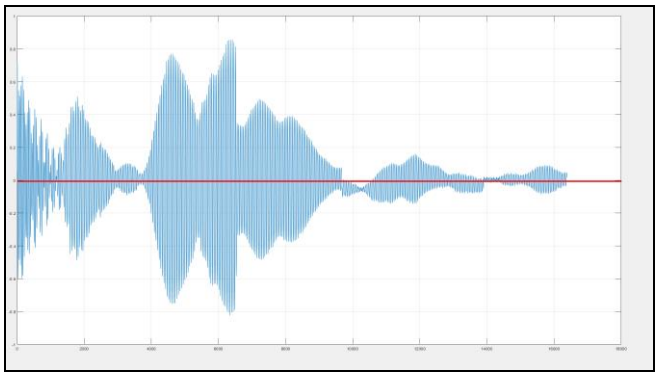


Figure 4: Pre-processed ADC signal with normalized values (from -1 to 1) and centered at 0 (red line)

In the next step, I decided to plot 100 samples of each class “human” and “ empty” in the same plot in MATLAB to investigate the likeliness of them, as shown in the Figure 5 below.

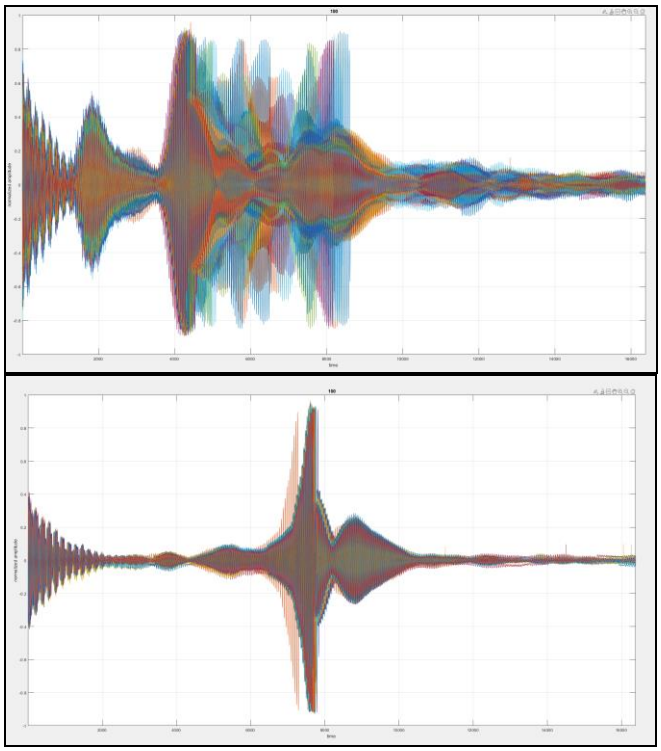


Figure 5: 100 refined ADC echo samples of classes “human” (above, SHC) and “empty” (below, EC) on top each other, showing the different convergence of each class in the collected data

For the “human” class because the human movements within the automobile cabin are unpredictable and arbitrary, there are a noticeable uncertainty in the middle range. However, if we consider the head and tail of the “human” signal, there is clearly a pattern that repeats through all the samples. It is curial to notice that in the “empty” class, the signal shape is converged into a little cone in the middle part of signal length, that shows there is a dominant frequency has occurred. This strong pattern will be a key to classify between two classes.

2) Gabor transformation

Until this point, it is feasible now to apply Gabor transform to each echo sample of two classes to produce the spectrograms. For this part of the project, I have reused some assets from a previous project "Discrimination of reflected sound signals by applying Gabor transform and CNN classifier". The implementation in MATLAB of Gabor transformation is hugely inspired by Mr. Nathan Kutz and his magnificent lecture in time-frequency analysis :" Inferring structure of complex system" by the department of Applied Mathematics at the University of Washington. [5]

To summarize the method in brief words and get a good understanding about the Gabor transform, it is viable to list out most critical steps in the process as following:

a) Generate Gaussian filter (window) with specific size and shape

b) Define the time step of the window, that will determine how rapid the window will be slided over the signal

c) Slide the filter over the signal, that requires to be analysis by Gabor transform

d) Multify the window with the signal in each step

e) Apply Fast Fourier Transform (FFT) to that signal within the window, to look for the most dominant frequencies and their spectral power contribution

f) After the window reaches the end of the whole signal, stacking all the FFT results to construct the spectrogram.

For more details, there is a deep explanation line by line in the program called as "prepare_data" can be found at my previous report.[3] An example of Gabor transform can be illustrated in the Figure 6.

In the Table 2, there are some important parameters that need to be taken care of during the Gabor transform. By changing these parameters, the user can affect the processing time to procedure the spectrogram, the feature sharpness in each of them, and their resolution.

Table 2: List of parameters I used to produce the spectrograms of two classes in the Gabor transform implementation

Parameter	Current value	Effect
Time step	0:100:16000	Progressing time, feature sharpness
Bell-shaped filter width parameter	$5*10^{-5}$	Progressing time, feature sharpness
Limit of Y-direction in TFR (frequency boundary condition)	[40,70]	Spectrogram image resolution
Spectrogram export dot-per-inch	50 DPI	Spectrogram image resolution

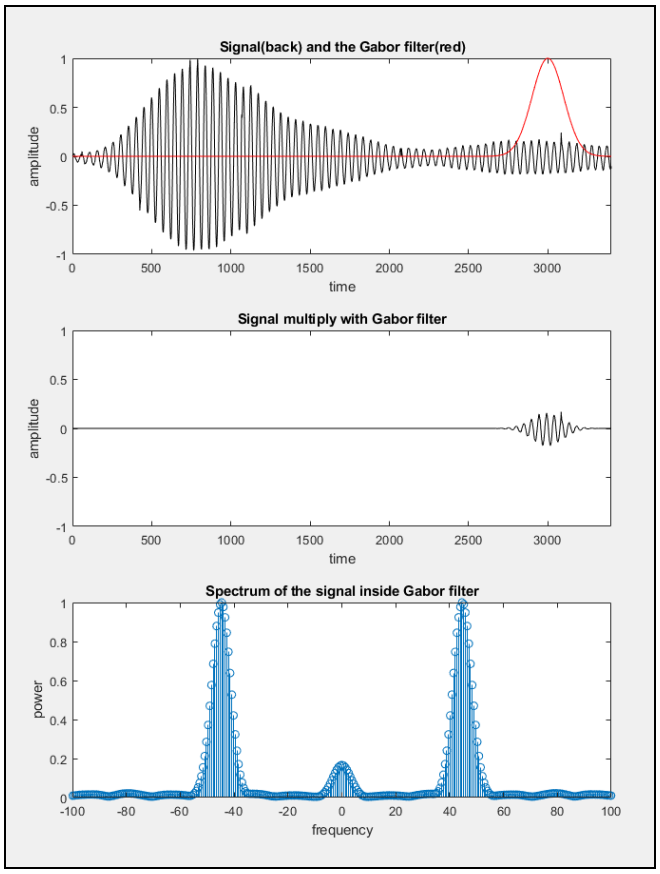


Figure 6: (top) Signal in black with red Gaussian sliding window; (middle) The signal multiplies with the window; (bottom) The spectrum analysis by Fast Fourier Transform of the only signal inside the window Notice: The signal is the figure is not the echo signal I used in this project. It is only meant to be an example for Gabor transform.

After applying Gabor transform, the result spectrogram of two class "human' and "empty" can be seen in the Figure 7 below. Even with our human eyes, we can clearly see the different feature of each image. For the "human", there are multiple spectral appearing over the image. On the other hand, for "empty", there is only a unique spectral in the center of the image. Adding many variations during recoding the input data, such as the door is shut or not, human is moving or stays stable, the seat belt is put on or off, is not affects much to this key feature.

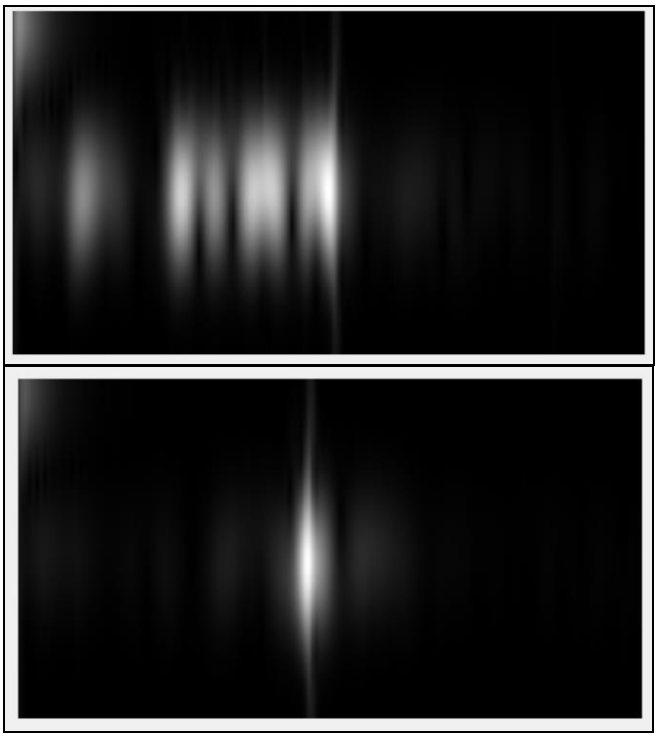


Figure 7: The 1st result spectrogram of "human" (SHC) and "empty" (EC) in the range between 40kHz to 70kHz respectively

D. Dataset build up

To create a dataset for the later CNN to train, it is essential to gather a good number of samples during data acquisition step. By the rule of thumb and multiple testing, I have come to the solution to use 1400 ADC echo samples (700 of human, 700 of empty) to build up the dataset. The amount is each class is identical to avoid unbalanced problem during training the network later. The dataset can be increased much further, but since the problem only to discriminate between only two classes, 1400 is sufficient in my opinion.

There are some examples of each variation of the input ADC signal showing in the following Figure 8 and Figure 9. The image size has been fixed to 361x456x1, to reduce the training CNN time in later step. The Table 3 indicates the portion of each variation in each class.

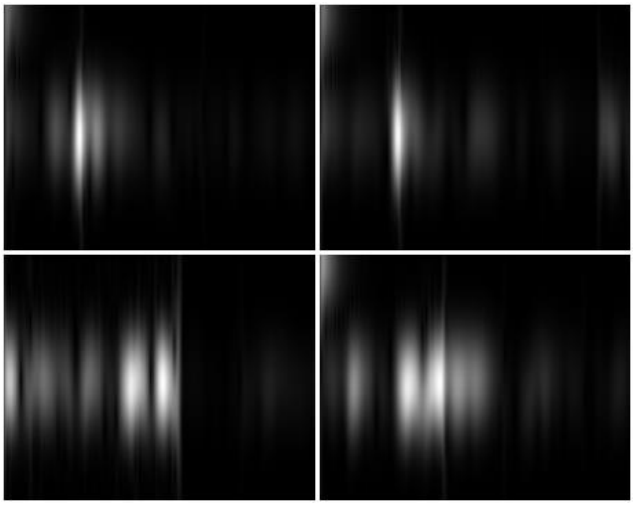


Figure 8: "Human" class, BHC, BSHC, HC, and SHC spectrograms (left to right, top to bottom)

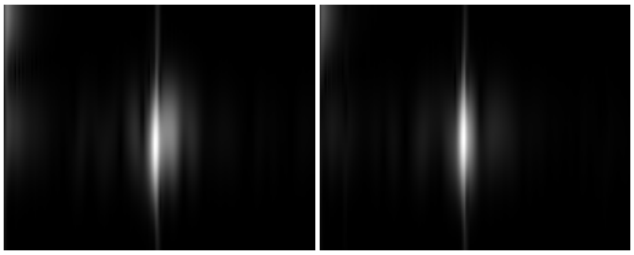


Figure 9: "Empty" class, BEC and EC spectrograms

Table 3: The contribution of variated ADC inputs to the dataset

Input ADC signals	Number of samples
BHC	300
BSHC	100
HC	100
SHC	100
BEC	300
EC	400

E. CNN architecture

The CNN model is used in this project is the sample CNN for the MNIST classification problem.[4] I have used the architecture as the foundation for the classifier for two class "human" and "empty". The architecture of the CNN model can be described briefly, as the following:

- Input layer matches with the image size (361x456x1)

- 2D Convolutional layer, filter size 64x64, 8 filters, same padding
- Batch normalizing layer
- ReLU layer
- 2D Max Pooling layer, pool size 32x32, stride 8

- 2D Convolutional layer, filter size 32x32, 16 filter, same padding
- Batch normalizing layer
- ReLU layer
- 2D Max Pooling layer, pool size 16x16, stride 16

- 2D Convolutional layer, filter size 16x16, 32 filter, same padding
- Batch normalizing layer
- ReLU layer

- Fully connected layer
- Softmax layer
- Classification layer (2 class)

F. CNN training

After training the CNN with multiple attempts, which has been shown in the Appendix at the end of this report, I have found the optimal size for the dataset. I used the gradient descent with momentum as the optimization algorithm during training the CNN model.

Firstly, I have used 400 images (200 “human” & 200 “empty”) to train CNN initially. The spectrogram resolution is 183x200x3 because I export the images still in RGB format. The CNN model still can perform acceptably with this small dataset, to classify between two classes. However, the model accuracy is fluctuating and kept unstable after 5 epochs. I acknowledge the small size of the dataset is the main problem to the performance of the model, thus I decided to increase the size of the dataset drastically.

For the second attempt, I have increased the dataset size from 400 to 1400 images, which include 700 images for each class. In the step of spectrogram generation, I used RGB to gray-scale conversion to reduce the image resolution from 183x200x3 to only 183x200x1. The CNN model accuracy during training is still around 75% and 85% even after 50 epochs. At this point, I address the problem with the spectrogram resolution. As the image size is small, the key features, which are the number of spectral over the spectrogram, might be blurry and undetectable by the CNN. Therefore, I decided to increase the size of the image and re-generate a new dataset to work with.

As the final attempt to train the CNN model, I used 1400 images with improve resolution of 361x456x1. With this new, improved dataset, the model performs well and reaches the final accuracy of 100% in classifying between “human” and “empty” classes.

I stored all the model parameters into a “CNNnet.mat” as the information the network have learned. To use the network to predict new coming ADC data, it is possible to reload these model weight again. To help the users to work my implementation better, I have designed a GUI, which will be explained clearly in the next part.

G. GUI design

1) GUI overal design

With the help of MATLAB APP Designer, I have made a GUI for the usage of classifying any new random ADC that the REDPITAYA sensor might produce in the future. In the UI, there are three important regions that the users need to pay attention to.

The first region is in the top of the UI, where the application requires input ADC text files from REDPITAYA sensor. The user can collect new data from the software, which is currently working, and specify the number of samples the sensor can export. Since I have used a specific function called as “readtable” in MATLAB, the input text file must have at least two rows of data, or two reading samples at minimum. If the input text file is successfully imported to the UI, then the program will pick randomly a sample and plot it out in the window in the left-hand side. The plot should have 16000 sampling points, which each has a normalized value, ranging from -1 to 1.

The second region is where the user can apply Gabor transform to produce the corresponding spectrogram of the chosen sample from previous steps. The image of the spectrogram if the UI runs correctly, will be presented in the right-hand side area, next to the plotting area of input signal.

The third region, which is the most important one, is at in the bottom of the region of the application. This part of the program will give the user the ability to export the images of spectrogram earlier into an image file (.png). There is a green lamp, which will alert the user when the exportation is completed. The image is essential for the later step of classification. The network weight will be reloaded to predict either a human appears in front of the REDPITAYA sensor or not, by indicating a red lamp.

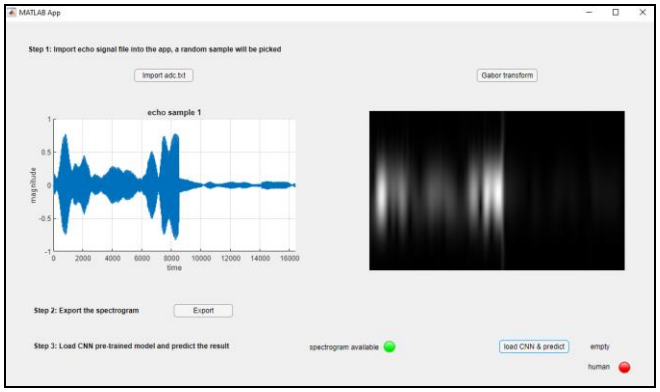


Figure 10: The designed GUI to predict new ADC signal base on the previous trained weight of CNN model

2) GUI user manual

Step 1: Put ADC output file from REDPITAYA sensor in the SAME directory as sensor_project_GUI.mlapp. Close all MATLAB program currently running.
Note: ADC file must have AT LEAST 2 rows of data, otherwise the app will not read it !

Step 2: Rename it EXACTLY to adc.txt, otherwise the app will not find it !

Step 3: Open sensor_project_GUI.mlapp (MATLAB with additional Toolboxes requires)

Step 4: Click on "Import adc.txt" button

Step 5: The left-side plot should show a random echo sample from the adc.txt.
Note: Please ignore the warning: "spectrogram.png is missing" by MATLAB

Step 6: Next, click on "Gabor transform" button.

Step 7: The right-side plot will show the spectrogram, resulting from the random echo sample respectively.

Step 8: Now, click on "Export" button.
Note: Please do not click or hover your mouse on the Gabor plot, otherwise it will distort the image file "spectrogram.png".

Step 9: Wait until the blue lamp is on, indicating that the image file is successfully exported !

Step 10: Finally, click on "Load CNN & predict" button, the prediction will be indicated by a red lamp.
Note: Ensure CNNnet.mat is in the SAME directory as sensor_project_GUI.mlapp.
Note: Run the application at first time will cost extra time to fully load the network.

V. TESTING & RESULT

A. *Testing*

To verify the effectiveness of my approach in classify between "human" and "empty" space within automobile cabin, I have decided to use a new set of ADC input signals, that have never been learned before by the CNN model. They are also recorded on a different day, under unlike conditions in compared to the training dataset. List of them is included as follow in the Table 4:

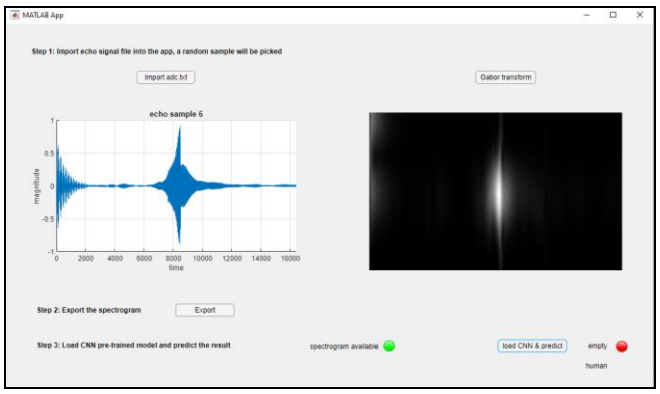
Table 4: New ADC input signals that CNN model never learns before

Code name	Meaning
BEO	seat **B**elt is put on, **E**mpty space with **O**pen door.
EO	seat **B**elt is put off, **E**mpty space with **O**pen door.
BHO	seat **B**elt is put on, **H**uman occupied with strong movement in the cabin space with **O**pened door.
HO	seat **B**elt is put off, **H**uman occupied with strong movement in the cabin space with **O**pened door.

B. *Result*

As the result, the GUI I have designed can successfully to predict correct classes, as shown in the Figure 11 and 12. Regards to the way I designed the GUI, the program can not process a batch of inputs, but on the other hand only one random input per trial. Therefore, it is not possible to create the confusion matrix. However, as I have intensively tested my implementation with considerable amount of input data, I have not encountered any mistakes made by my implementation yet. The processing speed is relative fast and depends on the running hardware (required a GPU to run) if we exclude the time of initial loading model.

All codes and dataset are stored online at [6], in my personal GitHub account, in case of any future developments of this project. A zipped version is also submitted to the examination board along with this report.



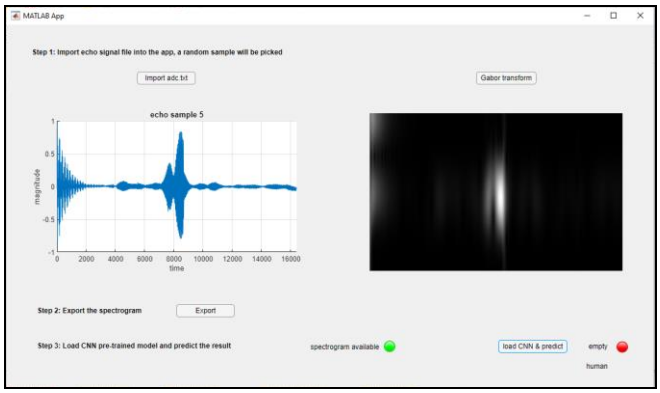


Figure 11: BEO and EO ADC input signals have been correctly predicted as "empty" by the GUI

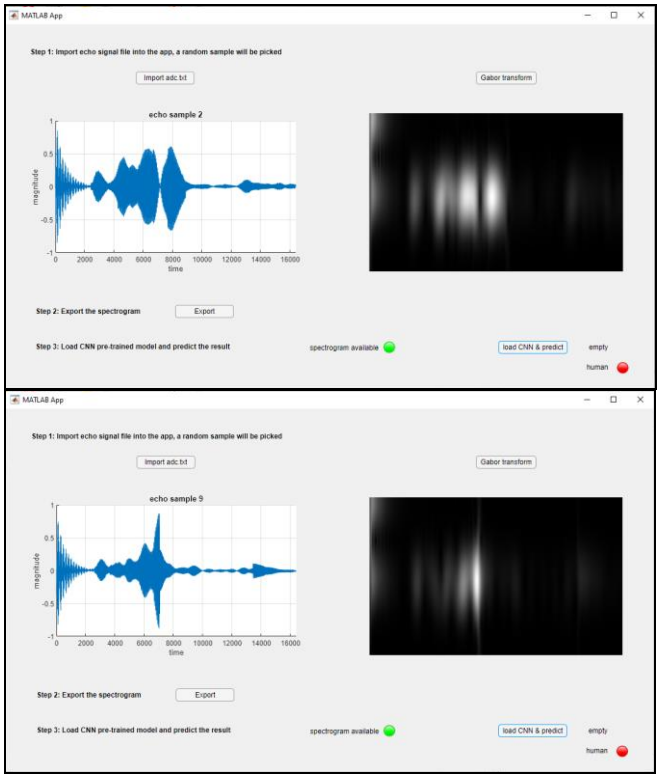


Figure 12: BHO and HO ADC signal have been correctly predicted as "human" by the GUI

VI. CONCLUSION & RECOMMENDATIONS FOR FUTURE WORKS

A. *Conclusion*

As the result of this project, it is plausible to conclude that my approach, in applying Gabor transformation and CNN classifier, predicts correctly and reliably when a human has entered a seat within the automobile cabin or not.

Firstly, I have designed the pipeline to receive the output text file storing all the ADC input signal of various scenarios from the REDPITAYA sensor board. Then the text files are converted into appropriate spectrograms, which have distinguished spatial features. To detect and use them as the key to classify between two classes, CNN model has been used with the help of a good number of images in my dataset. Finally, a GUI is designed at the end point to the future users and to validate the result as well.

B. Recommendations for future works

However, if considering the other possibilities to keep improving this project in the future, there are few points that I would like to call attention to:

- Increase the number of classes that are needed to be classified.
- Changing the sensor hardware to a better one.
- Moving from MATLAB to other programing language to unite with data acquisition steps with REDPITAYA, to form a uniformed pipeline from getting the sensor signal to class prediction.

ACKNOWLEDGMENT

I would like to express deeply my thanks of gratitude to my professor Prof. Dr. Peter Nauth, who gave me the golden opportunity to explore his fascinating project on the topic "Investigating ultra-sonic signals in discriminating different objects" which have intensively broadened my knowledge and practical experience in this field.

Secondly, I would also like to cherish our parents and friends who encouraged us a lot during this project to finish it within the limited time frame.

Thirdly, I want to send my thanks to Mr. Julian Umansky for his assistant with the REDPITAYA sensor board.

REFERENCES

[1] D. Gabor, "Theory of communication. Part 1: The analysis of information", Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering, vol. 93, no. 26, pp. 429-441, 1946. Available: 10.1049/ji-3-2.1946.0074.

[2] HoangLong, Nguyen, "Computational Intelligence: Audio Classification: CNN Network vs. LSTM Network – A Comparison".

[3] HoangLong, Nguyen ThanhDuy, Nguyen ThanhToan, Truong "Machine Learning: Discrimination of reflected sound signals by applying Gabor transform and CNN classifier".

[4] MathWork, "Create Simple Deep Learning Network for Classification" available at "https://www.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html".

[5] Nathan Kutz, "Time Frequency Analysis & Gabor Transforms" belongs the online course" Inferring Structure of Complex System" lecture [Online] available at https://www.youtube.com/watch?v=4WWvvMkFTw0.

[6] HoangLong Nguyen, "sensor_project" available at https://github.com/long2811/sensor_project.

VII. APPENDICES

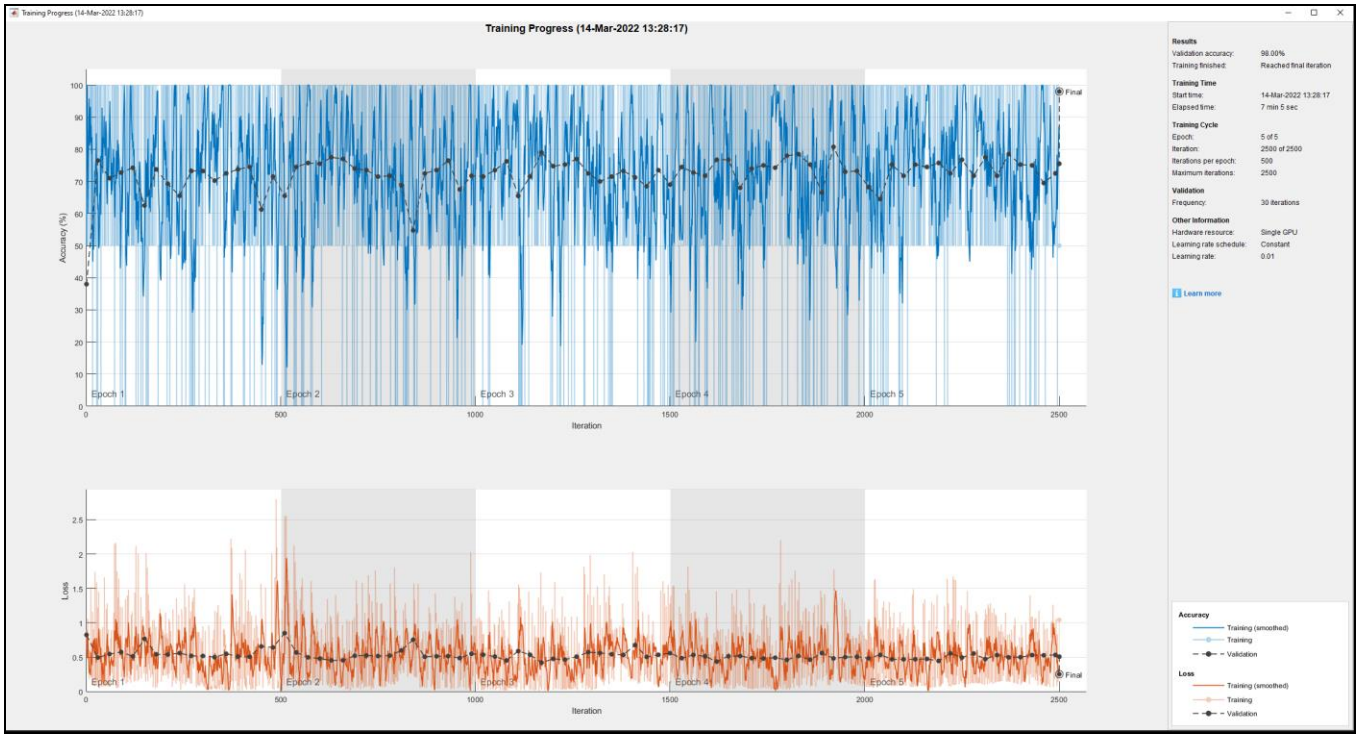


Figure 13: First training CNN with low number of images in the dataset (400) with low image resolution (183x230x3) with small epochs (5)

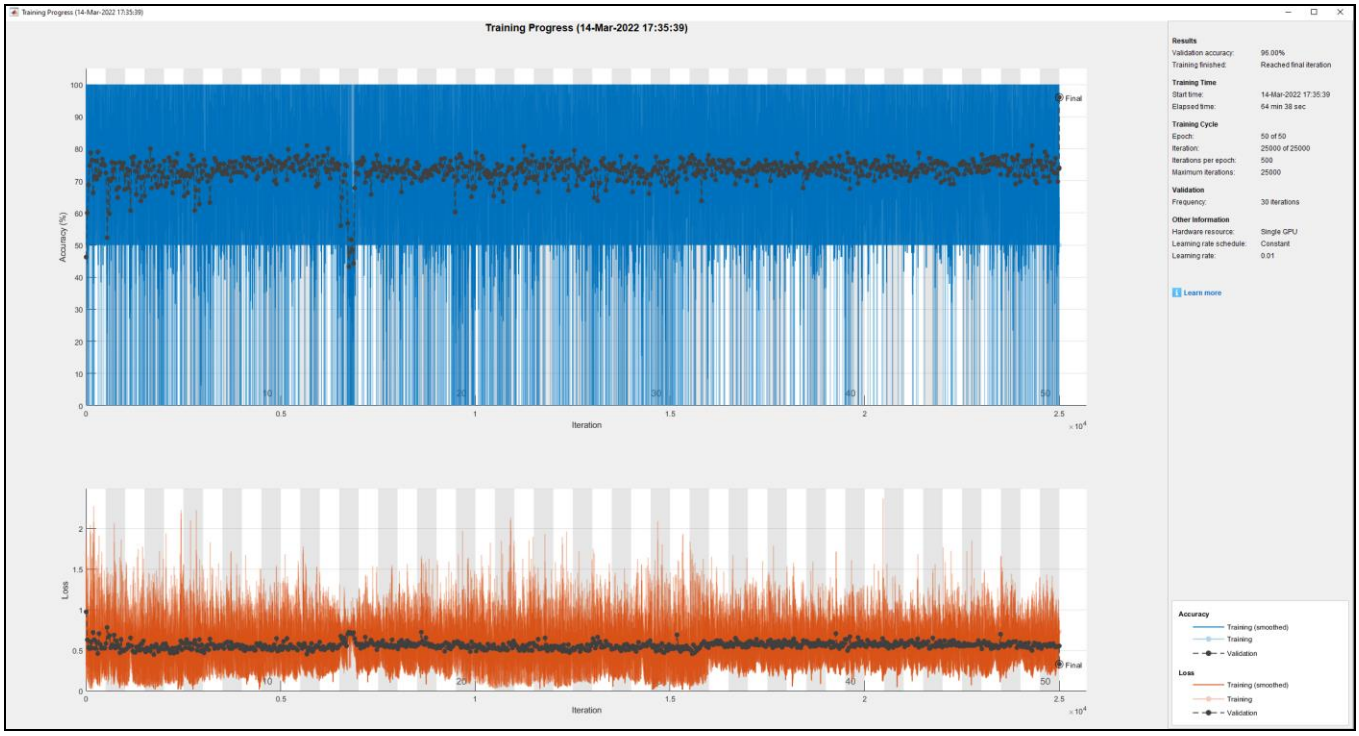


Figure 14: CNN model struggles still to classify between two classes after 50 epochs regards to high number of images (1400) in the dataset and low image resolution (183x230x1)

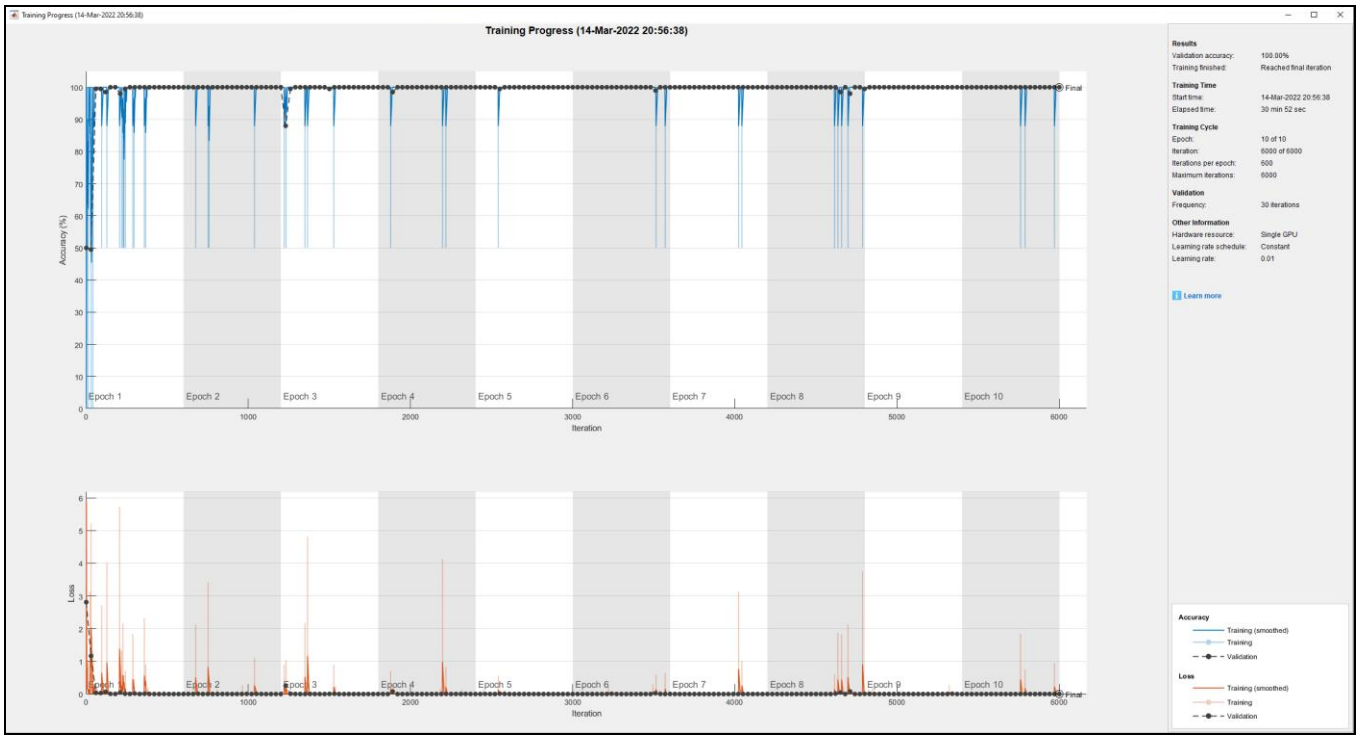


Figure 15: CNN model successfully classifies two classes with the same dataset (1400), but higher resolution images (361x456x1)

Figure 16: Final confusion matrix of classification of the improved dataset (1200 to train, 200 to validate)