# Application Notes

## EZTech - EESTech Challenge

## Our Solution

### Scope and purpose

In this project we had to do a Machine Learning Algorithm that detects the number of people moving in an indoor room using data provided by Frequency-Modulated Continuous Wave (FCMW) Radar.

### Intended audience

We hope with this project to implement a simple algorithm that could count the number of people inside an indoor room. We did a algorithm that could easily be run in any recent , so anyone with the radar and a computer can run this program.

## Table of contents

# 1 Project

For this project we decided to use a Convolutional Neural Network (CNN) to classify the data we gather from the FCMW Radar. We decided on a CNN due to its simplicity and great compatibility with 2D data. Also, the CNN can be easily run in almost every recent computer/smartphone due to its low computational effort, we can see that by the time that the prediction takes, 24ms.

During training the data, we achieved an accuracy of 77,25%, we know that this doesn't correspond to the real-world accuracy due to many factors, like overfitting, different places that we gather the data, the small amount of data that we gather compared to what is needed (we had to gather our own data, so we didn't had time to build a huge dataset), etc.

Also, we decided to use some out of the box ideas and implementation, we will talk more about these ideas in the following points.

## 1.1 Gathering and handling the data

A big part of this challenge was the data gathering, since none was provided to the participants to train and test the models. The conditions where the data was gathered were not ideal but good enough to see promising results.

So, as to gather the data from the sensor, we used the pre-processing function provided by Infineon. In addition, we added some lines of code for data processing, as seen in the following table:

**Code Listing 1:**

```
001            for k in range(0,data_epoc):
002                for j in range(0,4):
003
004                    display_text("STOP - NEXT: " + text[j])
005
006                    time.sleep(1)
007
008                    display_text(text[j])
009
010                    for i in range(0,f2r):
011                        raw_data.clear()
012                        hit = 0
013                        for i_frame, frame in enumerate(device):
   # Loop through the frames coming from the radar
014
015
   raw_data.append(np.squeeze(frame['radar'].data/(4095.0)))      #
   Dividing by 4095.0 to scale the data
016                            if(i_frame == number_of_frames-1):
017                                data =
   np.asarray(raw_data).astype('complex128')
018
019
   range_data=processing.processing_rangeDopplerData(data,True)
020                                #range_data=np.abs(range_data)
021
022                                for n1 in range(3):
023                                    for n2 in range(number_of_frames):
024                                        aux = aux +
   np.abs(range_data)[n2,n1,:,:]; #average of the 10 frames
025
026
   data_dump.append((aux/number_of_frames))
027                                #print(np.shape(data_dump))
028                                break
029
030                        ts=calendar.timegm(time.gmtime())
031
032                        hit=hit+1
033
034
035                    print("finished frames")
036
037                    print(np.shape(Y_train))
038                    print(np.shape(np.ones((f2r), dtype=float)*j))
```

**Code Listing 1:**

```
039
040                     if np.size(Y_train) > 1 :
041                         Y_train = np.append(Y_train, np.ones((f2r),
   dtype=float)*j)
042                     else:
043                         Y_train = np.ones((f2r), dtype=float)*j
044
045
046                     print("training")
047
048                     #Neural Network
049
050             X_train=data_dump
```

In theses lines of code, we basically gather the data from the sensor and store the Doppler Data in the *X_train* variable, with this data we can train our model. We decided to use the data that we gather directly in the model, training it in real-time.

We decided on this real time approach due to the great size of the data needed to be used, it would be simpler and more effective since there would be no need to store this data and load it repeatedly. As such, we train the model in real time for quicker and less memory demanding computing. This is similar to a calibration and can be used to better adapt for each case.

Also, we store all models so we could use in another time without the need to train the model again, we have a program for that purpose (**load_model.py**).

For the data that we gathered to be useful, we had to label it, so we decided to do a program that presented the number of moving people that had to be in front of the sensor, as can be seen by the following picture:

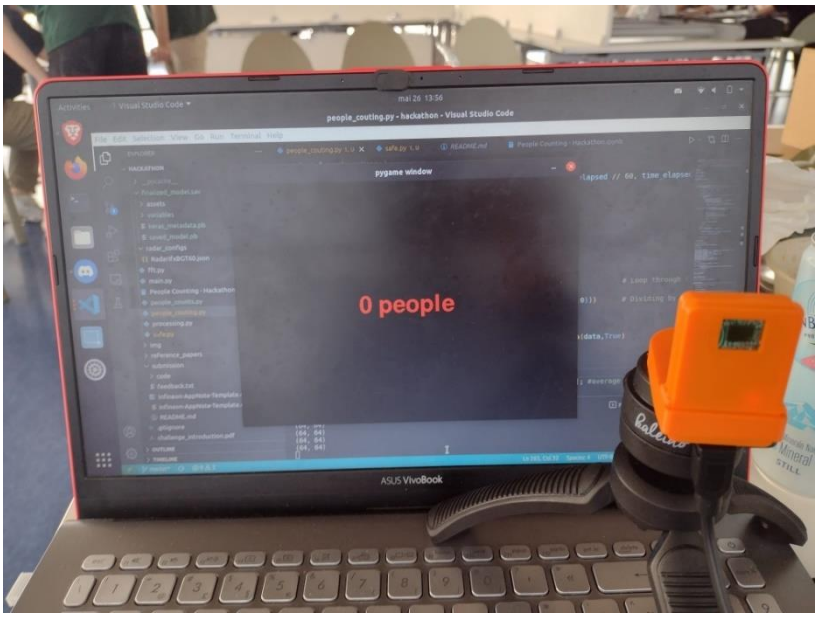


Figure 1 – Display showing how many people should be in front of the sensor during data acquisition

In addition, we had to introduce a timeout so the next person could come in on time and the data wouldn't be badly labeled. So, for this purpose we present the following message on the screen:

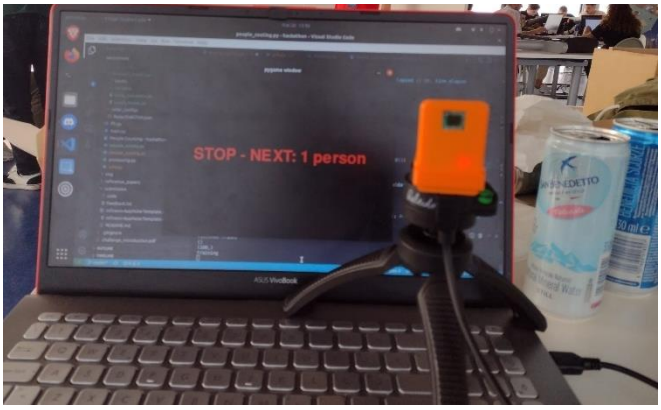


Figure 2 – Display showing how many people will need to be in front of the sensor in the next stage of data acquisition

Also, like everyone else we had to be our on-test subjects so when the program told that we had to have one person moving in front of the sensor, one of us had to be there, like we can see in the following image:

Figure 3 – Data acquisition process

## 1.2 Convolutional Neural Network

For the Machine learning model, we used a CNN, as priorly mentioned. As said before it is very simple and effective to use. It is also very not very resource demanding compared to other options, as we can see by the 24ms that took to predict the number of people in front of the radar.

The input data to this model was a 64 by 64 by 1 ***numpy*** array. As we were using this type and dimension of data. We decided on using a 2D convolutional layer.

### 1.2.1 CNN Structure

The CNN was developed using the ***tensorflow*** library, with special use of ***keras***. We decided on this due to its simple implementation, versatility, and the team's experience with the library.

The CNN created is sequential and composed of 2 convolutional layers of 128 with kernel sizes of (4,4), with hyperbolic tangent (*tanh*) activation before an average 2D polling layer. It is then followed by 2 more convolutional with 64 with the same kernel sizes and activation.

After this, its followed by a max polling and flatten layer, and then by a 2 fully connected 1028 layers and finally a 4-neuron layer with *softmax* activation for classification.

The code of this CNN is as followed:

**Code Listing 2:**

```
001            model = models.Sequential()
002            model.add(layers.Conv2D(128, (4, 4), activation='tanh',
   input_shape=(64, 64,1)))
003            model.add(layers.Conv2D(128, (4, 4), activation='tanh'))
004            model.add(layers.AveragePooling2D((4, 4)))
005            model.add(layers.Conv2D(64, (4, 4), activation='tanh'))
006            model.add(layers.Conv2D(64, (4, 4), activation='tanh'))
007            model.add(layers.MaxPooling2D((4, 4)))
008            model.add(layers.Flatten())
009            model.add(layers.Dense(1024, activation='tanh'))
010            model.add(layers.Dense(1024, activation='tanh'))
011            model.add(layers.Dense(4, activation='softmax'))
```

### 1.2.2 Model Saving and Loading

Every CNN developed is saved for posterior use and testbenching.

We save the CNN by using the following code.

**Code Listing 3:**

```
012        model.save(filename)
```

We also developed a program solely for loading and running a model to see its "true" performance (**load_model.py**).

This is used to test a model not by the accuracy obtained with a dataset but for a real-world applicability.

## 1.3 Results

### 1.3.1 Data Results

For the models developed, most obtained good results for the data acquired during the data acquisition process, with the highest accuracy obtained during validation of 77,25%.

There is, however, a great difference between the capability of obtaining great results with virtual stored data and obtaining results with real world and real time captured data. The “virtual” results do not reflect the real-world applicability.

There are a lot of factors that explain this difference such as poor conditions when acquiring data:  people walking behind the defined area and other sensors being used close by. Also, the constraint of not being able to obtain large amounts of data make it difficult to generalize to most cases.

### 1.3.2 Real World Applicability

Even with the constraints, the results show that a solution based on the work done during this hackathon is possible, since there were models capable of distinguishing fairly well the number of people.

## Revision history

| Document version | Date of release | Description of changes |
|---|---|---|
| 1.0 | 26/05/2022 | Added Solution and Project Chapters |
| | | |
| | | |

**Trademarks**
All referenced product or service names and trademarks are the property of their respective owners.

**Edition yyyy-mm-dd**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**



**Do you have a question about this document?**

**Email:** erratum@infineon.com

**Document reference**
**AppNote Number**

**IMPORTANT NOTICE**

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

**WARNINGS**

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.