**ML application for counting people**

**Team B2+**

**About this document**

**Scope and purpose**

The purpose of this document is to provide an insight to our solution for the problem of counting people using a 60GHz Infineon radar.

**Intended audience**

This document is intended for everyone involved in the final round of EESTech challenge in Milan, 2022, both contestants and jury.

[**Data collection and preprocessing**](#_3znysh7) **4**

[Data collection](#_2et92p0) 4

[Preprocessing](#_3txfrpyyffrn) 4

[**ML approach**](#_1xf5l31h877p) **5**

[Keras CNN](#_y0l1wcsjcgkh) 5

[Training and validating](#_pxvh0w55siqv) 5

[**Algorithm complexity**](#_skng5hjfk7mb) **6**

[Preprocessing](#_geqc5b5a87y9) 6

[Real time processing](#_et8l8jlcwmf6) 6

[**Key points**](#_kckz2x7lidig) **7**

[Passing values between frames](#_z1up47kb2fz5) 7

[Relative accuracy](#_k6hxsv6vja2n) 7

[**Dependencies**](#_xm009846a016) **8**

# Data collection and preprocessing

## Data collection

Required data to train the ML model was collected using *ifxdaq recorder* software which was provided in contest materials. We recorded individual material for each of 4 situations:

-no people

-one person

-two people

-three people

Each category was properly labeled after recording and saved for training purposes.

## Preprocessing

Radar settings implied data format which contains frames, each frame containing 64 chirps and each chirp consisting of 128 samples. Using FFT we convert this data to a range- doppler plot which is 64x64. By precise measuring, we were able to cut the right half of the plot (>3m range) and the top and bottom quarters (very high velocity, impossible for people to move that fast), thus leaving us with 32x32 format data.

# ML approach

## Keras CNN

Our ML approach of choice was using a Keras sequential model. After evaluating many different approaches, we decided to choose this one because our data was formatted in a similar way as images and we had enough knowledge and experience to process it as such. With certain preprocessing and preferred data formatting, this approach yields sufficiently good results regarding accuracy and loss.

## Training and validating

All the data from our custom made datasets was randomly shuffled and split into two categories, one used for training and the other one for testing purposes. For the first one, we took 80% of labeled frames from the data pool, leaving 20% for evaluating and validating the model. A sufficient number of test runs and demo models left us with the right parameters such as epoch number, which helped us maximize accuracy. In the current version of the model, measured accuracy is around 85%.

# Algorithm complexity

## Preprocessing

Preprocessing explained earlier in this document provided a way to cut off unused data and minimize model complexity and working time, while keeping the useful data structured and with minimal memory occupied.

## Real time processing

For demonstration purposes we set an arbitrary limit for testing only every third frame. This is not necessary, the model is efficient enough to work with every frame, however the GUI displaying the results needs to run on a separate thread to avoid stuttering.

# Key points

## Passing values between frames

We developed a strategy to minimize the impact noise and false results have on the final count. To do this, we pass the previous detected value to the next frame. As the number of people grows, due to the angle and distance limitations, the accuracy falls (people overlapping or covering each other). While this doesn’t affect the zero people and one person results, the accuracy on 2 and 3 people scenarios deteriorates quickly. Because of that, zero value adds nothing to the next frame, one value adds 5% chance to 1, two add 10% and three add 15%. We measured accuracy with and without the improvement and we got 15% more accurate results with the improved version.

## Relative accuracy

For even more precise measurement, we included a concept we call “relative accuracy”. This means that we don’t show only the absolute accuracy of the highest rated class, but also the absolute accuracy of other classes and the difference between two highest rated classes.

# Dependencies

-Python 3.9

-Tensorflow

-Keras

-Numpy

-tkinter

-PIL

Note: all dependencies are free to use and available via pip

**Revision history**

| **Document version** | **Date of release** | **Description of changes** |
| --- | --- | --- |
| 1.0 | 27.5.2022. | Initial commit |