Detecting Moving People with radar sensor

by HAM(Maria Boukoutsou ,Afroditi Letsiou,Chrysa Lampropoulou)

About this document

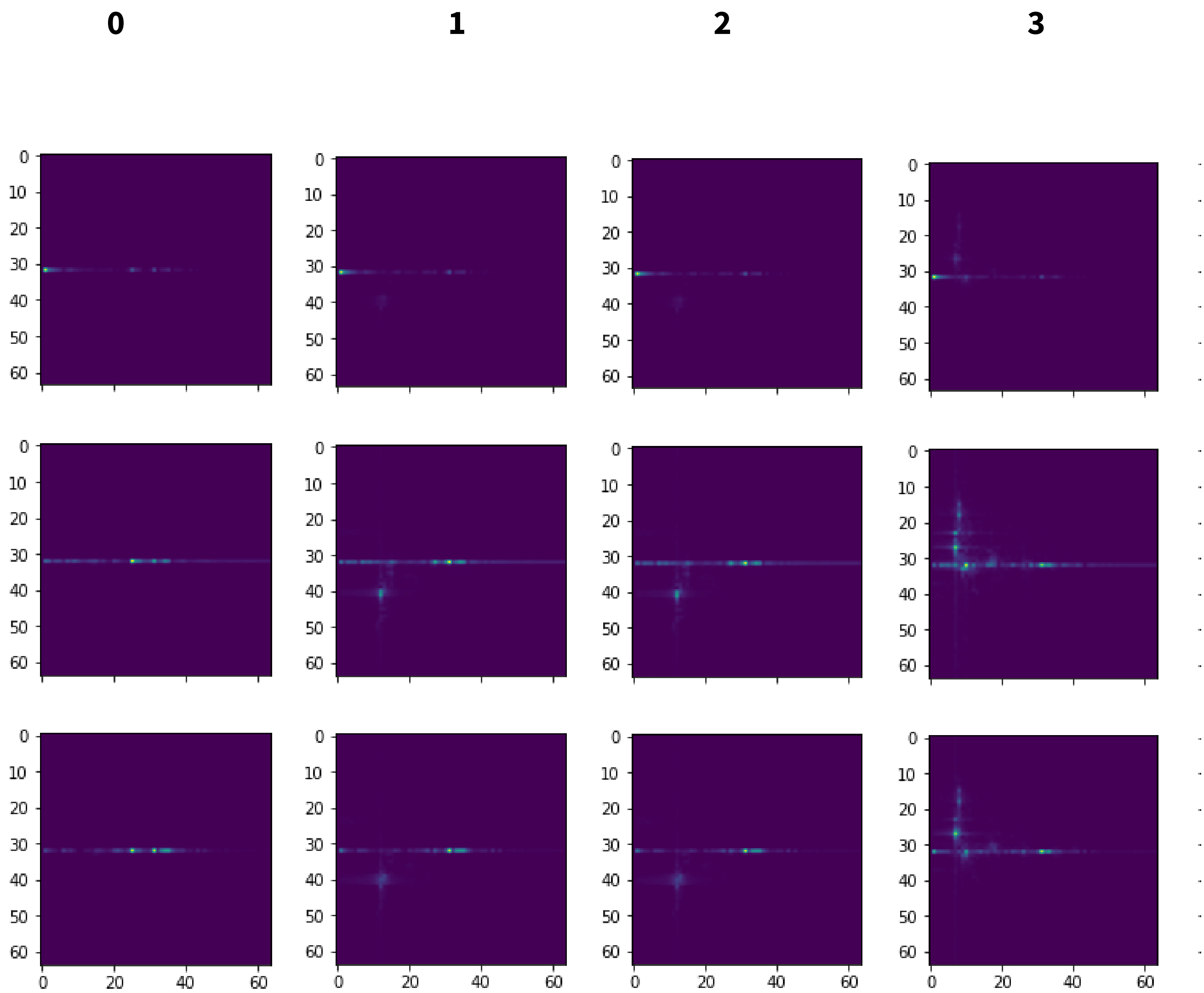
Scope and purpose

The purpose of this project was to build a machine-learning-based People Counting solution using Infineon's BGT60TR13C radar chip in an indoor environment. It can be applied in cases of 0,1,2 or 3 people in the room and the height of the radar from the ground should be between 0.8-1.5m.

Table of contents

About this document 1
Table of contents 1
1 Collection of the Data and Preprocessing .... Error! Bookmark not defined.
2 Deep Learning 4

1. Collection of the Data and Preprocessing



To create our training dataset, we collected a sum of 3000 frames for each one of the following situations:

- existence of no moving person in the room
- 1 moving person
- 2 moving people
- 3 moving people

To begin with, we used the given code for the collection of the data.

We converted the raw_data variable to an array of shape (3x3000 , 64x128) and then created our target variables which consisted of a column with zeros for the first situation and so on for the rest.

We, then, appended the information for the target variable to the dataframe and saved the data to csv file.

At this point, we have one large dataset with the 4 different situations, so a dataset of shape (4x3x3000, 64x128).

This is an unnecessarily big dataset, since some information is being repeated because of the existence of the 3 receiver antennas. We thus, kept only 1/3 of the information to avoid wasting computational resources.

Making the dataset

```
copyarray0 = raw_data

copyarray0 = np.array(copyarray0)  # FOR NO PEOPLE

copyarray0.shape = (9000,64*128)

ones = np.ones((9000,1))*0

final0 = np.append(copyarray0,ones,axis=1)


df0 = pd.DataFrame(final0)

df0
```

	0	1	2	3	4	5	6	7	8	
0	0.466911	0.438584	0.419536	0.422955	0.434188	0.429792	0.410256	0.401465	0.413675	0.4263
1	0.519658	0.498168	0.469841	0.461538	0.477167	0.491819	0.484005	0.464713	0.454945	0.4671
2	0.487912	0.468864	0.459585	0.474969	0.494261	0.487912	0.470330	0.468376	0.481563	0.4908
3	0.468376	0.440537	0.420757	0.421001	0.432234	0.429304	0.411722	0.401709	0.413187	0.4271
4	0.518926	0.499634	0.470330	0.460073	0.476679	0.493284	0.485958	0.465446	0.454212	0.4664
...	...	...	...	...	...	...	...	...	...	...
8995	0.516239	0.488889	0.460073	0.458852	0.482540	0.494750	0.480586	0.463004	0.461538	0.4752
8996	0.480830	0.463980	0.465446	0.483516	0.492308	0.480342	0.465690	0.469841	0.485470	0.4903
8997	0.459096	0.433700	0.423932	0.431746	0.437118	0.424420	0.406349	0.405372	0.422466	0.4307
8998	0.515263	0.487912	0.458608	0.459096	0.482051	0.494261	0.480586	0.461538	0.458608	0.4722
8999	0.479853	0.463248	0.464957	0.482540	0.492063	0.479609	0.466911	0.470818	0.485470	0.4903

9000 rows × 8193 columns

2. Deep Neural Network

To create our prediction model, we used a deep learning algorithm, a Convolution Neural Network, since we needed spatial information to be taken into account.

After we imported our training dataset, we kept the last column as the target variable and the rest as the features. We scaled our data using MinMaxScaler() and transformed it by multiplying it with 255, since the data is visual. We also did a random train-test split in our dataset (67% training-33% test).

Convolution Model

Then we started building our model. We added some convolutional layers with kernel size(3,3). We also added pooling layers and a fully connected layer.

ANN

At this point, we added 3 more layers with activation functions 'relu' and 'softmax' since we are aiming to multiclass classification.

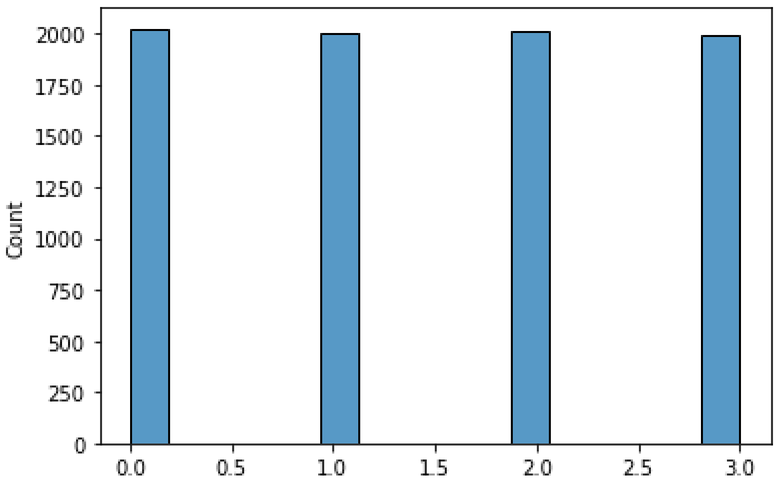
Also, because the classes are 4 (either 0, 1, 2 or 3 people in the room), the last layer has 4 neurons.

We then compile, using as the loss function *'sparse_categorical_crossentropy'* and the adam optimizer.

2.1 Validation

In this part of the algorithm, we applied the validation data to the trained model and took an array in which each row contains the probability that there are 0, 1, 2 or 3 people in the room accordingly. We selected, then, as the model's prediction the number of people that maximizes this probability, in order to compare it with the actual number of people and evaluate our model.

In the diagram below, we can see the distribution of our target variable in the training set.



Evaluation

We can notice that the accuracy at the training set is very close to that of the testing set, a sign that the model is not overfitted.

When, on the other hand, we move on to the validation phase, we see the accuracy decreasing since the model has learnt to predict the class in the specific spatial environment of the training-testing set and is, thus, having a hard time predicting the correct number of people in a new space.

If we were validating with data collected from the same space, the accuracy of the model would be much better

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