False Data Classification in radar point clouds using DNNs

Jagyan Prasad Mahapatro[1] and Mahdi Chamseddine[2]

[1] yagya.mhp@gmail.com
[2] mahdi.chamseddine@dfki.de

Abstract. This report presents the results of various experiments performed in identifying false data in radar based point clouds. We see the problem statement in the introduction where we look at the potential of point cloud data, the advent and usefulness of radar data and the challenges of using radar point clouds instead of lidar based point clouds. We look at different components of the project, i.e., the radar sensor and the model for training. We present the ground truth preparation strategy, data preparation and training strategies implemented in the project. After that we test the model and implement different strategies to improve the accuracy of the model. In the experiments section we see the results in the form of tables and graphs and then see some qualitative results as well. Finally in the conclusion section we look at what are future challenges in the model improvement.

Keywords: Point Clouds, radar, lidar, PointNet [2], Point-wise classification, False Data Identification, False radar points

1 Introduction

By increasing the accuracy of 3D sensors such as RGB-D, Time-of-Flight and lidars, 3D information is showing effectiveness in many applications such as 3D localisation, robots navigations and autonomous driving systems. In this paper we are considering only one kind of 3D information that is point clouds.

1.1 Point clouds

Point clouds are the simplest representation of 3D shapes and scenes which are usually generated by lidars. They provide a rich geometric structure and they prove to be advantageous where only visual information is not enough to extract feature information. Hence, the underlying structure of the points provide a better insight into the objects of the scene which can be helpful in many different applications such as autonomous driving, indoor navigation, robotics, etc.

1.2 radar based point clouds

But lidars have certain drawbacks like being bulky in size and very costly. Also in bad weather lidars do not produce accurate results. Thus, there has been a need of a cheap and small 3D sensor which generate point clouds which could be mounted and manufactured practically. Radar sensors do fill this need as they are small and cheap along with being robust to weather conditions. In recent years, they have been modified to generate point clouds by using their functionality of object detection. radar is a detection system that uses radio waves to determine the range, angle, or velocity of objects.

1.3 Challenges with radar based point clouds

However, there are a few issues which come in using point cloud data generated by radars. The radio waves have a tendency to be reflected on metallic surfaces. Hence when recorded, the location of the point clouds is read falsely as can be seen in Fig. 1 where we see in comparison to radar there are many false points. So, to effectively use the point clouds generated by radars, we have to be able to distinguish the false points from the right points. Further, the right points could be used for further tasks like classification and segmentation. In this project, we try to use deep learning to do this. As this task is attempted for the first time, we choose PointNet [2] [] as the network as it is a simple network and it does point-wise processing. In this report we shall look at the methodology used to achieve this and then in the evaluation section we shall see the results achieved through experiments.

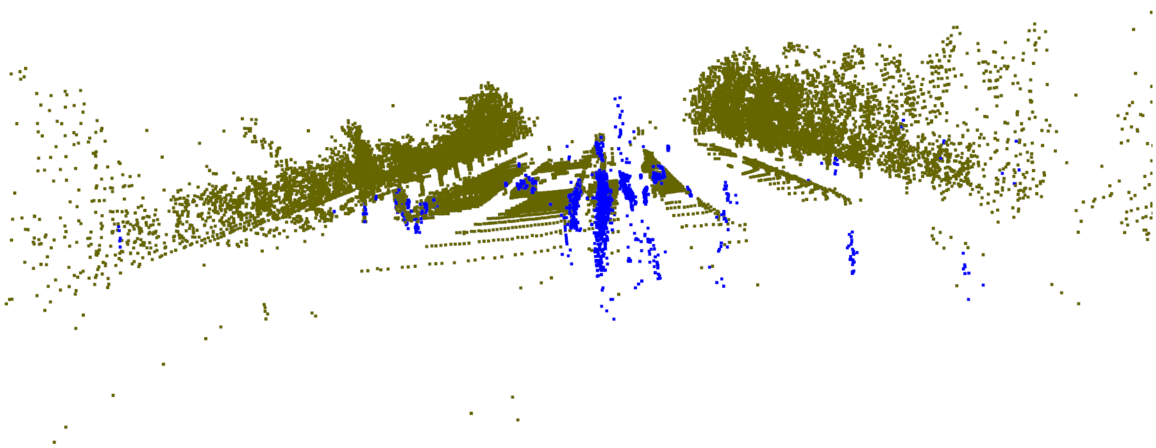

Fig. 1: **Pointcloud example**. Olive colored point are lidar points and blue points are radar points.

2 Components

The project consists of two major components, the data and the network. In this section we shall look at the type of radar sensor used to capture the data and the network which is used to classify the points in the point clouds as true and false points.

2.1 Frequency-Modulated Continuous-Wave radar (FMCW radar) [3]

The data used in this project is gathered using FMCW radar. FMCW radar (Frequency-Modulated Continuous Wave radar = FMCW radar) is a special type of radar sensor which radiates continuous transmission power like a simple continuous wave radar (CW-radar). In contrast to this CW radar FMCW radar can change its operating frequency during the measurement: that is, the transmission signal is modulated in frequency (or in phase). Possibilities of radar measurements through runtime measurements are only technically possible with these changes in the frequency (or phase). The basic features of FMCW radar are:

- Ability to measure very small ranges to the target.
- Ability to measure simultaneously the target range and its relative velocity; Very high accuracy of range measurement.

– Signal processing after mixing is performed at a low frequency range, considerably simplifying the realization of the processing circuits.
– Safety from the absence of the pulse radiation with a high peak power.

2.2 PointNet [2]: Architecture and Evaluation

Most approaches which work with point clouds usually transform them into other structures such as Voxel grid or Mesh structure. But, PointNet [2] is a unified architecture that takes point clouds directly as input and classifies whole input as a label or assigns labels to each point. In the initial stages the points are processed as separate data points and hence its architecture is very simple. It needs only the three x, y and z coordinate value as the input but more features such as normals, intensity or colors could be provided as additional inputs to improve results.

The network architecture is visualized in Fig. 2. Here, it can be seen that the classification and segmentation networks share much of the portion of the architecture. The pipeline is explained in the caption of Fig. 2. We can see that this architecture takes into consideration the local feature of the point which includes the x, y, z and other additional features and then the global feature which information about a point as part of the whole input. The point features are aggregated through max pooling which is useful in handling the unordered nature of point clouds.

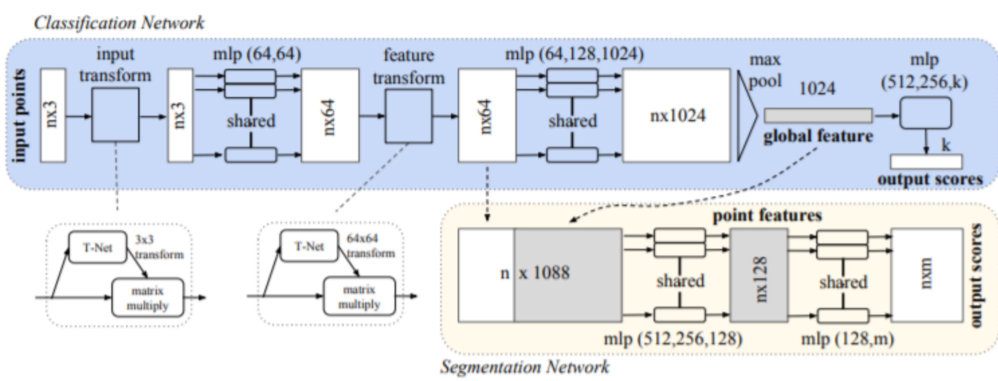

Fig. 2: **PointNet [2] Architecture**. The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

3 Project Approach

The implementation of the PointNet [2] segmentation network for the task of false radar data classification consists of a few steps.

3.1 Ground truth preparation

One of the primary challenges in data preparation was to find a way to assign labels to points in radar point clouds to have as ground truth for the training purposes. Because of the large number of points, doing this manually is not very time efficient. So to do this, we utilised the lidar point cloud of the same scene. Considering lidar sensor to be accurate, we check that if a radar point has a lidar point within 1 metre of it. If it does, we consider the point to be a true point else a false point. The range is considered to be 1 metre because that is the error range of the radar sensor. To do this efficiently, kd tree algorithm was implemented to find the nearest neighbor.

3.2 Data preprocessing

There is some calibration information with respect to the radar sensor which need to be combined to the radar data. For this, a calibration matrix associated with each data set is multipled with the radar point cloud. Moreover, a limit of 60 metres is put on the dataset as the lidar data used to prepare the ground truth has a range of approximately 60 metres.

3.3 Model modification

A very good advantage of using PointNet [2] is that it is a very simple network which can accommodate any number of features as input and with very little modification, the number of output classes could also be controlled. We are using the segmentation network for our task as we need point wise labelling. We used the k dimensional part as input of the model setting k to be 5 as we have 5 dimensions associated with each point, XYZ values, Velocity of the point and Magnitude of the point. For the output, we need only true of false for each point. SO there is only one class with probability of belonging to this class is between -1 and 1. To assign the class, all values more than and equal to 0 are classified as true points and those below 0 are classified as false points.

Also, the input to the model is 2500 points but most scenes in the dataset contain much less than that. Hence in the dataset module of the code, while sampling the points, if a set contains more than 2500 points, then we sample the required out of them but if they do not, we fill the remaining with zeros as they would not contribute to the model.

3.4 Model evaluation

The evaluation of the model is done using the test dataset, which consists of 80% of data. We just calculate the number of correctly classified points with respect to the total points of the test dataset to evaluate the accuracy of the model. At this point, the zeros appended to the points of a scene to have 2500 points have to be disregarded as they do not belong to the dataset and were generated to match the model.

However there was a problem in using the random split of dataset between train and test sets which we shall see in the next module in improving the accuracy.

3.5 Steps to improve accuracy

The results of evaluation are listed in section 5 but a few steps were taken to improve the accuracy of the model.

- **Changing the learning rate**: The default learning rate of the model was 0.001, we experimented with learning rates 0.01 and 0.0001. There was however no significant improvement in accuracy.
- **Modification of data splitting**: Initially, we were using a random 80-20 train-test split of the dataset but out of the 6 measurement sequences, 4 were static and did not contribute much to the training, hence only the remaining two dynamic ones were used for training.
- **Data Augmentation**: One of the main challenges of the project was the limited size of the dynamic dataset. So to have more data for training, rotation(between -5 and 5 degrees) and translation(between -1 and 1 metre) was implemented to double the dataset.
- **Weight decay**: There was a case of overfitting on running the training for more than 50 epochs. To avoid this, weight decay was implemented. Weight decay[1] is defined as multiplying each weight in the gradient descent at each epoch by a factor smaller than one and greater than zero. This technique is equivalent to introducing an L2 regularization term to the cost function that one wants to optimize. This penalizes the complexity of the model and hence avoids overfitting.

4 Experiments

The experiments and results are divided into three sections. We look at the dataset parameters where we look at the size of the data, the parameters and the division of the dataset into different datasets. After that we look at the different parameters of evaluation that we use and finally we look at the results before and after implementing the improvement strategies. We also look at qualitative results in the form of point clouds being segmented.

4.1 Dataset parameters

The dataset is captured using the radar sensor that we discussed in section 2. The data is in the form of XYZ, Intensity and Velocity if the object that the point belongs to. Using these five features we try to train the model to classify the true and false points. This data is divided into seven measurement sequences. Two of the sets are dynamic measurement sequences and five are some what static. Parameters about the measurement sequences are stated below. We shall also look at some sample of the sequences.

- **Dynamic sequences**: There are two measurement sequences in training set with 2,338 scenes and 373 scenes respectively. An example of a scene is as below.

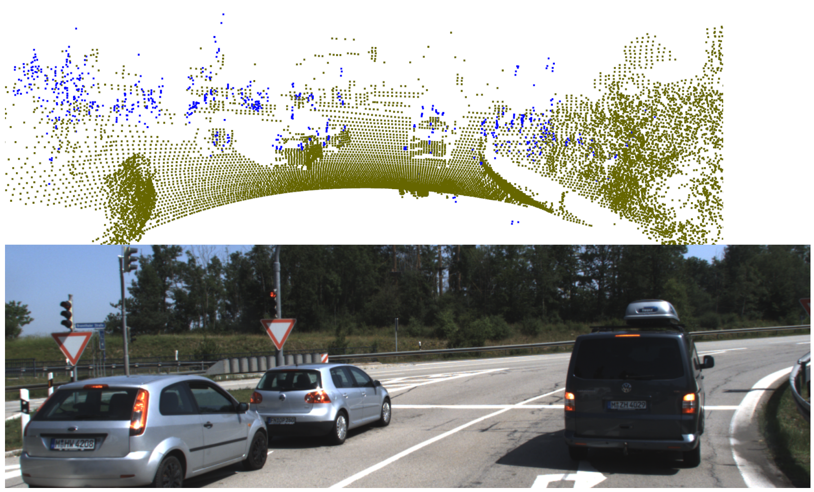

Fig. 3: **Dynamic data example**. Olive colored point are lidar points and blue points are radar points and the image for the corresponding scene.

– **Static sequences**: There are four measurement sequences in test set with 682, 549, 222, 689 and 635 scenes respectively. A sample scene is presented below.

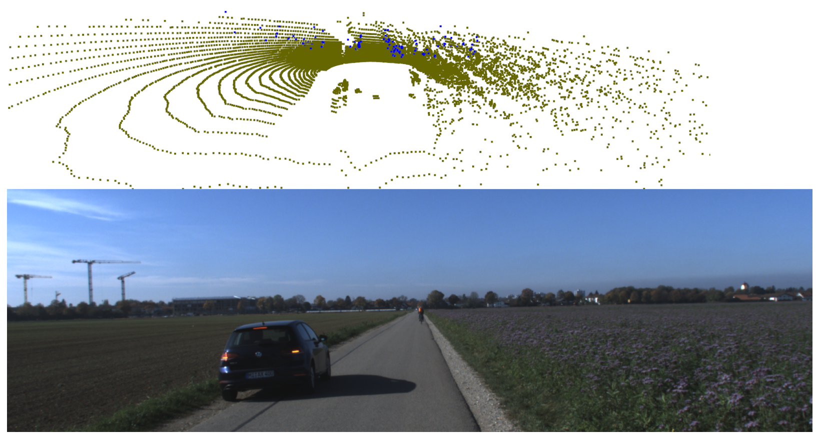

Fig. 4: **Static data example**. Olive colored point are lidar points and blue points are radar points and the image for the corresponding scene.

4.2 Evaluation parameters

To evaluate the results we look at the following parameters for both training and test measurement sequences:

– **Accuracy**: This is just in the form of percentage for the correctly classified points with respect to total points. This just provides quantitative results for the data.
– **Loss**: As there are only two classes which we are trying to predict, we are using binary cross entropy as the loss function. Binary cross entropy [1] measures how far away from the true value (which is either 0 or 1) the prediction is for each of the classes and then averages these class-wise errors to obtain the final loss. The loss value is presented in the following formula:

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=0}^{N}(y * \log(\hat{y}_i) + (1-y) * (1-\hat{y}_i))$$

– **Quality of Results**: Finally in the qualitative results sections, by visually inspecting the results we try to find out that if all the important points are being classified correctly. All the improvements made in the model have also been made looking at the quality of these results.

4.3 Results

We shall look at the various results of our experiments in three ways. We shall look at a confusion matrix for the ghost points and then a table showing IoU values at different stages of training. Finally, we shall look at a few sample results and then discuss the effectiveness of the model for both dynamic and static measurement sequences. The number of points sampled for this network is 2500 points.

	Predicted: True	Predicted: Ghost
Actual: True	28.39%	7.81%
Actual: Ghost	3.51%	60.29%

Table 1: **Confusion Matrix.** This percentage is calculated on total number of points in the test dataset with classes being TRUE and GHOST points.

Confusion matrix: A confusion matrix is a summary of prediction results on a classification problem. The number or percentage of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. If we look at the confusion matrix in Table 1, we find out that most of the ghost points are identified correctly. This shows that the task that we are trying to achieve in this project.

Accuracy:

Method	Training Accuracy (%)	Test Accuracy (%)
Random Split training	89.4%	79.1%
Dynamic-Static Split training	94.7%	85.13%
Data Augmentation	94.8%	85.17%
Training with weight decay	94.8%	89.17%

Table 2: **Accuracy at different stages of training.** This percentage is calculated on total number of points in the test dataset with classes being TRUE and GHOST points.

Qualitative Results: The qualitative results are divided into two figures. Fig. 6 shows sample results in some static test scenes and Fig. 7 shows sample results in some dynamic test scenes. We can see that as scenes get more and more dynamic, the quality of results deteriorate. This shows that with more training with dynamic data, the results would improve.

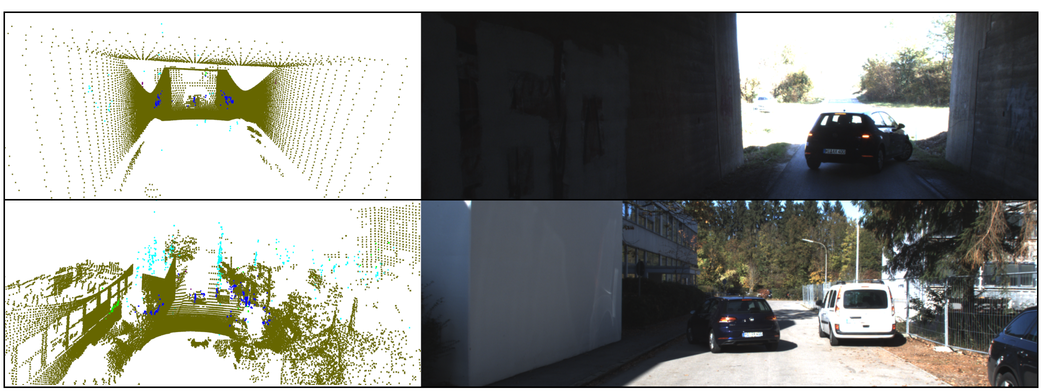

Fig. 5: **Static scene results**. Olive colored point are lidar points, blue points are correctly classified true radar points, green points are incorrectly classified true radar points, teal points are correctly classified ghost radar points and purple points are incorrectly classified ghost radar points.

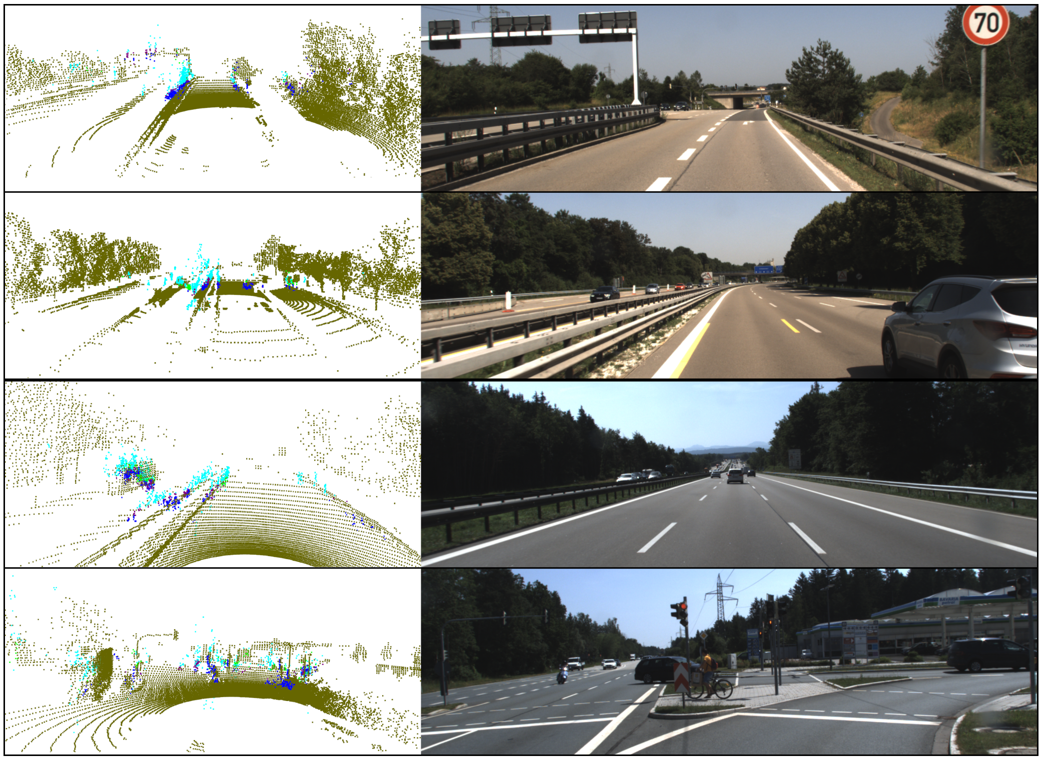

Fig. 6: **Dynamic scene results**. Olive colored point are lidar points, blue points are correctly classified true radar points, green points are incorrectly classified true radar points, teal points are correctly classified ghost radar points and purple points are incorrectly classified ghost radar points.

5 Conclusion

This network performs well for being the first attempt at detecting ghost points in radar point clouds. Although the network used for the task, PointNet [2], is a very simple network,it was able to achieve very good results. There were a few challenges which limited in achieving even better results most important of which was lack of more dynamic data. As we see in section 4.1, we have many scenes available but the number of dynamic scenes available is less. This even led to overfitting in the beginning of the project. Weight decay was implemented to get rid of that. Data augmentation was implemented to increase the training data but that did not improve the results by much. The point clouds generated by radar sensors are sparse even by point cloud standards as we can see in Fig. 1, thus the only way accuracy could be improved is by putting more dynamic scenes into training.

This task has a lot of potential as there is a very real way that radar point clouds could become part of navigation and autonomous driving in the future and with the advent of better quality radar point clouds the results of this project could be drastically improved.

References

1. Peltarion. Binary crossentropy. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy, 2020. [Online; accessed 10-March-2020].
2. Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In *the IEEE conference on computer vision and pattern recognition*, 2016.
3. Christian Wolff. Frequency-Modulated Continuous-Wave Radar (FMCW Radar). https://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.html, 2019. [Online; accessed 10-March-2020].