Radar Object Classification Using Neural Networks in Urban Automotive Scenarios

Atila Gabriel Ham
Communications Dept.
Politehnica University of Timisoara
Timisoara, Romania
atila.ham@student.upt.ro

Corina Nafornita
Communications Dept.
Politehnica University of Timisoara
Timisoara, Romania
corina.nafornita@upt.ro

Vladimir Cristian Vesa
Forvia Hella Romania
Timisoara, Romania
vladimircristian.vesa@forvia.com

***Abstract*—This paper proposes a method to solve the problem of object classification for automotive radar systems. This is done using neural networks and knowing attributes about the objects such as their trajectory, acceleration, velocity, velocity and acceleration standard deviation, object length and width and other parameters related to object's movement. The method proposed is validated by experiments, obtaining in process of validation an overall accuracy over 99%: for the pedestrian class an accuracy of 99.88%, for the cyclist class an accuracy of 99.23% and for the car class 99.82% accuracy.**

Keywords—neural network, feature, radar, accuracy, activation function, loss function, layer, velocity, acceleration, standard deviation for acceleration.

I. Introduction

Autonomous driving requires that cars equipped with radar sensors as well as other sensors, need to be aware of their environment, but also be able to understand it. Thus, the task of classification of moving object types that surrounds the car is very important. Several papers address this topic. Classical papers in target recognition identify radar data features and use a machine learning classifier such as support vector machines (SVM) [1] [2] [3]. In [4] the authors proposed a deep learning classifier, which uses radar reflections as input, where each reflection is characterized by its range, radial velocity, radar cross section, and azimuth angle. In [5] the authors apply spiking neural networks (SNN) to automotive radar object classification. In [6] the authors propose a method based on LSTM (Long short-term memory) layers which consist of several steps: data is translated into a common coordinate system, grouped, and labelled before being provided to the classification model.

The classical radar architecture begins with the antenna that receives the signal, after that it is converted from analog to digital. The signal processing block has the main objective to output the raw targets. Next, the targets created will be sent to the tracking module which will be outputting an object. Here the main objective is to output the object as precisely as possible and to classify it correctly (e.g. cyclist, motorbike, pedestrian, car). The goal of this paper is to propose a method to classify the moving objects for automotive radar systems, in an urban scenario, by using neural networks. They will be classified in three categories: pedestrian, cyclist and car. The features used are object's width, length, speed, acceleration, standard deviation for speed, acceleration. Here all the data (exception is the standard deviation for position on X and Y axes) are expressed as the speed and velocity overground and not as a relative velocity. It is important to take just those features that will be able to discriminate between different object classes, thus the velocity could be considered as a discriminative factor. A pedestrian has a significant smaller velocity compared to a car or a cyclist, a car has a non-uniform way of moving in an urban scenario due to traffic, with lots of acceleration and deceleration, such that the velocity and acceleration standard deviation will be higher than for a pedestrian or a cyclist which will have a more uniform way of moving in urban scenario.

The paper is organized as follows. In Section 2 we present the proposed method, in Section 3 we give the implementation details while in the last two sections we show the results as well as conclusions and future directions.

II. Design Methodology

The working flow diagram of the proposed method is shown in Fig. 1.

The first step is to choose representative features that will be extracted while running the application in Software in the Loop (SIL). It's important to have features that can make a difference between different types of object classes. For example, the length of an object can be defining, if we want to distinguish between pedestrians and cars, because a pedestrian will be significantly smaller than a car. The standard deviation for acceleration can be a good indicator if an object is a pedestrian or a car, because in an urban scenario, cars will have a higher standard deviation of acceleration due to their capacity to accelerate as well as decelerate faster than a pedestrian. It also indicates that a car will have a wider range of speed compared to a pedestrian or a cyclist.

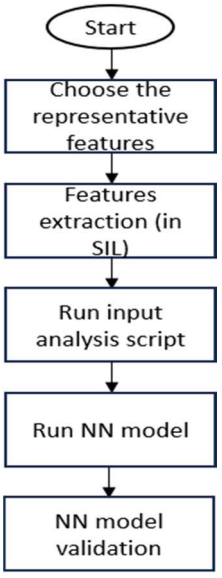

Fig. 1. Workflow diagram.

The second step is to run certain traces, which can be recorded in a controlled environment or can be free tracks, recorded in public space, the latter being preferable, and the

979-8-3503-9086-5/24/$31.00 ©2024 IEEE

extraction of these features from certain categories of vehicles. At this point it's important that these objects are real, clearly visible, not obstructed by other objects or obstacles, that they are not ghost objects. The obstacles could be for example ISO poles.

The third step is to run the feature analysis script on the dataset, obtained in the previous step, and rank it according to its impact on the classification of different objects. The purpose is to see which feature is more significant and will have a larger impact in the final classification decision.

The fourth step is to run the neural network and see what the output of that training will be. If the result is good enough (accuracy over 99%), then we can stop the training and make a step forward to the model validation. This step will consist of taking another dataset and running the trained NN to observe the results.

III. Implementation

In this section, we will present the implementation of the method proposed.

A. Dataset

The dataset will play a major role in the process of training the neural network and implicitly in the final decision. It's important to have a well-defined dataset, that will have recordings with objects in different situations scenarios in order to ensure a large diversity of data. Our dataset contains measurements from real objects, and the records were made when the host-vehicle was freely driving in an urban scenario.

There were 70 features in the current dataset possible candidates for training. The following features were selected for training:

- velocity,
- velocity on the x and y axes,
- acceleration,
- acceleration on the x and y axes,
- standard deviation of velocity on the x and y axes,
- standard deviation of acceleration on the x and y axes,
- correlation for velocity, acceleration and position.

As mentioned previously, velocity can be a differentiating factor for distinct classes, so a pedestrian's speed is lower than that of a car or a cyclist, and a cyclist's speed is lower than that of a car. Acceleration is another crucial feature since a pedestrian and a bike cannot accelerate as quickly as a car.

Another important feature is the size of the object, so width and length are important attributes which can differentiate between different objects. The estimated length and width of the object can be affected by several factors, such as the impossibility the see the full body of object when it approaches the car from the lateral side and we will have just a few detections only from a part of the object. This can be considered as a physical limitation of the system.

The present dataset was created by running various traces and comprises 18414 records. It was stored in CSV format.

B. Feature Analysis Script

A Python script will help in the process of deciding which feature should be used from the existing 70 features. It is to be noted that some features will be discarded because they contain predefined data or irrelevant data for classification, such as the object age or the number of radar cycles.

This script will read the dataset and will apply a scaler, MinMaxScaler, on the existing values in order to have all the data within a range from 0 to 1. This is done to avoid the neural network to rely on features of high values and to obtain a consistency in training.

The next step is to calculate the importance of the feature. For each object class, the mean of each feature will be calculated. After that, we compute the absolute value of the difference between the mean obtained for the current class and the mean obtained for the other classes (since there are three classes, there are three potential combinations). Following this step, we compute the mean of each feature across all three new classes. After that, we scale all the columns after the mean to get a mean column value ranging from 0 to 100. This will represent the feature importance by mean value.

Another possibility is to compute the rank for every feature. Rank is better to indicate the impact of every feature in the final classification because it can show the feature contribution, or how the feature will evolve, in classifying two different objects. In Fig. 2, the results of running the analysis script are shown. Classes are denoted as follows: 1 means pedestrian, 3 means cyclist and 5 means car.

Feature	1-3	1-5	3-5	Mean	Ranks
rcs	71.7637	150	78.2363	100	6
length	50.9817	140.684	89.7022	93.7892	8
classHighest	72.6151	120.86	48.2448	80.5733	9
xyVelAbs	19.5813	65.8411	46.2598	43.894	20
stdDevYawRate	25.8782	51.8747	25.9965	34.5832	21
fusionInfo	30.4214	42.1661	11.7446	28.1107	31
corrYPosOrientation	8.074	29.086	37.16	24.7733	44
width	5.17907	142.046	136.867	94.6975	46
corrXPosOrientation	25.1651	5.78532	19.3798	16.7767	50
corrXPosXVelocityAbs	24.0552	16.8064	7.24877	16.0368	54
existenceProbability	24.8604	29.9334	5.07303	19.9556	56

Fig. 2. Feature ordered by rank (last column). A feature is more important if its mean has a higher value and respectively its rank has a lower value.

On the first column we have the features considered, on columns two, three and four we have the absolute difference of the mean values between two different classes, 1-3, 1-5, 3-5, on the next column we have the mean, and on the last column we show the rank. For column two, 1-3, we have the absolute difference of mean value of class 1 (pedestrian) and mean value of class 3 (cyclist). For width feature, the result is a low value of 5.179065 which indicates that the width for pedestrian and cyclist is quite similar and it's not a definitory factor to classify them. On the third column, 1-5, for width feature, the value is 142.046278 which means that the width in this case is a good indicator to separate pedestrian and car. On the fourth column, 3-5, the value is a large one (136.867213) and indicates that this feature can be useful to separate the class car from the class cyclist. The mean column is a mean value of all the three columns. Width is in second place as the most important feature to distinguish between all the classes. Finally, the last column is a rank computed based on the mean value. This shows the feature importance and can give us a clear view on the contribution of each feature in the final decision on the classification process. For example, if we consider length (position 2 in Fig. 2 – rank 8) compared to

width (position 8 in Fig. 2 – rank 44), we can observe that the length is more important and can be considered as more efficient to distinguish between different object classes compared to width, since width is ranked lower in Fig. 2.

C. Neural Network

In the designing process of the neural network, the first step is to find its hyperparameters. In order to do this, we should implement a grid search and as the neural parameters we will use batch size, epochs, dropout rate, optimizer, learning rate and the hidden units. The best overall accuracy of 99.9674% was obtained using a batch size of 64, a dropout rate of 0.1, 50 epochs, 10 neurons, a learning rate of 0.01, and the RMSprop (Root Mean Square Propagation) optimizer. RMSprop was chosen since it is ideal if the computational effort needs to be small and the input data does not have a large variation. After the best combination of those parameters are known, we can start to use those results in our neural network.

The neural network has an input layer of type flatten, a hidden layer which consists of two dense layers, and an output one of type dense. The hidden layers have 10 neurons each and the output layer has the same number of object classes with 3 neurons. For hidden layers the activation function ReLu was used, and for the output layer SoftMax function was used. Also, the sparse categorical crossentropy was used as a loss function.

In order to balance the dataset used for training, Synthetic Minority Over-sampling approach (SMOTE) was used to generate synthetic data, and to ensure a consistency between different runs, a seed was set. SMOTE solves the issue of class imbalance in our classification task [7]. Class imbalance occurs when one class in the dataset (the minority class) has significantly fewer occurrences than another (the majority class).

For the dataset partitioning it was used 60% for training and 20% for validation and the rest of 20% was used for testing. The results will be considered valid only if the accuracy is higher than 99% for the validation part. If the accuracy for each class is lower than 99%, we will restart the training process. Also, the early stopping method was used in order to reduce the probability of overfitting the neural network, with a patience of 5 and the monitoring feature val_accuracy. ReduceLROnPlateau was also used in order to reduce the learning rate when we are close to a plateau on the learning curve, in this way avoiding the possibility of overshooting the model. For this a factor of 0.1 was set and a patience of 3 was chosen.

The features for network training are:

- corrXAccelerationAbsYAccelerationAbs,
- corrXPosXAccelerationAbs,
- corrXPosXVelocityAbs,
- corrXPosYPos,
- corrXVelocityAbsXAccelerationAbs,
- corrXVelocityAbsYVelocityAbs,
- corrYPosYAccelerationAbs,
- corrYPosYVelocityAbs,
- corrYVelocityAbsYAccelerationAbs,
- height,
- stdDevHeight,
- stdDevLength,
- stdDevWidth,
- stdDevXAccelerationAbs,
- stdDevXPos,
- stdDevXVelocityAbs,
- stdDevYAccelerationAbs,
- stdDevYPos,
- stdDevYVelocityAbs,
- xAccelerationAbs,
- xVelocityAbs,
- yAccelerationAbs,
- yVelocityAbs,
- length,
- width,
- xyAccAbs,
- xyVelAbs,
- stdDevxyAccAbs,
- stdDevxyVelAbs.

Those features are related to the object size and movement; they are given as absolute overground sizes, without considering the relative ones, which have as reference point the ego car. The data is split into features and labels. We apply scaling for the features, using the MinMaxScaler which is more suitable for data that does not have a normal distribution. The data is also shuffled, to ensure a random sequence for each training run, avoiding any underlying bias or order in the data.

IV. Experimental Results

The first step for the experimental results is to run the script for feature analysis. Although the script will use all of the characteristics in the prioritizing process, only those that will provide information on the object motion and dimension will be used.

Results for feature classification script are shown in Fig. 2. After we have a clear view of the features and their impact on training and validation processes, we will need to start the training process. For this the dataset will be divided in 3 parts, 60% of dataset will be used for training, 20% for validation and 20% for test. The validation process will be finished only when we will obtain an accuracy for each label over 99%, otherwise the training process will be repeated until we will reach at least 99% accuracy.

Fig. 3 shows the result after the training process (blue color) as well as validation process (orange color). In the validation process, we obtain for label 1 (pedestrian) an accuracy of 99.88%, for label 3 (cyclist) an accuracy of 99.23% and for label 5 (car) 99.82% accuracy. The highest score was obtained for pedestrian class, 99.82% of the pedestrians are correctly classified in the validation part of the model. To achieve this result, the training process ran four times, since in the first run the pedestrian class accuracy was 98.86%, below the required threshold of 99% for each label. In the second run the cyclist class accuracy was 98.93% and in third run the pedestrian class accuracy was 98.86%.

It is to be noted that there is a gap between the training and validation curve, due to the dropout rate of 1 neuron in each hidden layer. Using a dropout rate in the training process will give the neural network the ability to generalize the input values as long as the input values do not differ too much from the training values used and will make it more robust.

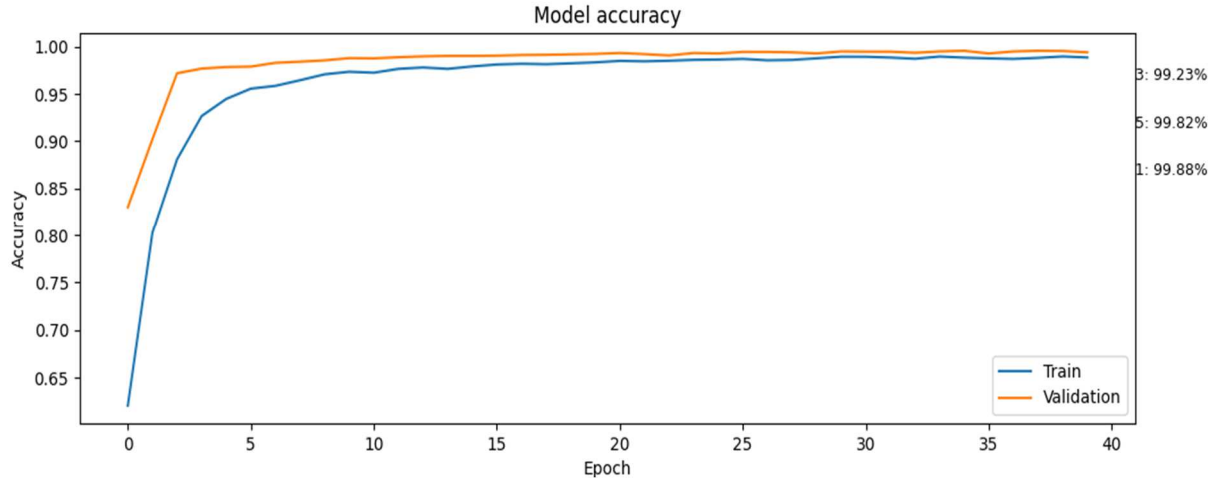

Fig. 3. Training and validation results.

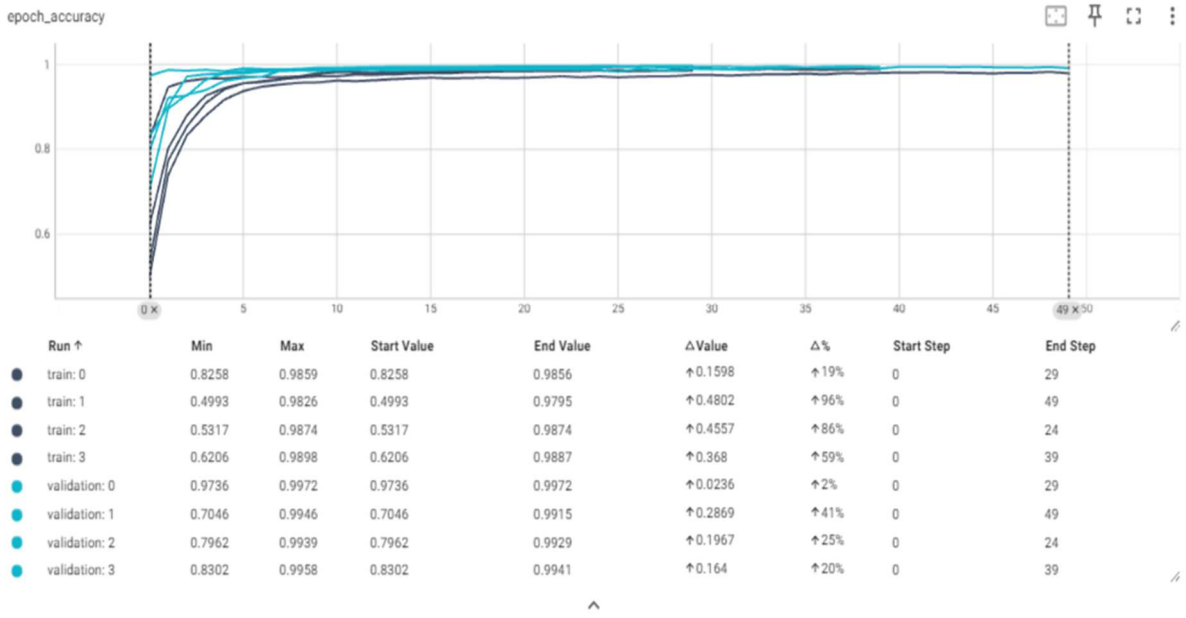

Fig. 4. Evolution of accuracy.

Fig. 4 shows the evolution of accuracy as a function of number of epochs. We can see that four training processes (dark blue color) and two validation processes (light blue color) took place to achieve satisfactory results. In all of the first three running the learning process was stopped due to reaching the plateau in the curve of learning and the possibility to overshoot the learning process.

Next step is to use the remaining 20% of dataset in order to see how the model will perform on the new data.

Results are shown in Fig. 5. For label 5 (car) we obtained an accuracy of 99.86%, for label number 3 (cyclist) we obtain an accuracy of 99.36% and for label number 1 (pedestrian) we obtain an accuracy of 99.58%.

The overall results are satisfactory and demonstrate the idea that if we know data about the urban travel mode of different classes of objects, we can predict the class to which they belong.

```
Accuracy for each label on test data:
3: 99.36%
5: 99.86%
1: 99.58%

Confusion Matrix:
[[1404    9    0]
 [   2 1411    0]
 [   3    3 1407]]
```

Fig. 5. Test for NN.

V. Conclusions

The aim of this paper was to extend the knowledge in the field of automotive radar by using neural networks to classify different types of objects in several categories: pedestrian, cyclist and car. The contribution of this paper is to consider automotive object classification only by knowing their width

and length and data about their way of moving, such as: velocity, acceleration, standard deviation for position on X axes and Y axes and so on. It starts from the premise that cars in urban scenario will have a higher velocity and a nonlinear way of movement compared with the pedestrian or cyclist which tend to move more linear. We have verified that the method proposed is successful by presenting experiments on a vast database and the neural network was capable of distinguishing between different types of objects in an urban scenario. We obtained for the pedestrian class an accuracy of 99.88%, for the cyclist class an accuracy of 99.23% and for the car class 99.82% accuracy on the validation process.

Once the test part begins, it was shown that the accuracy will remain quite stable and will not have a significant drop. The limitation of the current implementation is that once the scenario will be changed and we will consider an extra urban scenario the accuracy in classification will drop. As a future improvement we should also consider other features for classification, such as: signal-to-noise ratio (SNR), radar cross-section (RCS) and to start the investigation from the targets associated with the object and from there to extract any correlation between detection and the object class.

ACKNOWLEDGMENT

The authors thank Forvia-Hella for the support.

REFERENCES

[1] S. Heuel, and H. Rohling, “Two-stage pedestrian classification in automotive radar systems,” in 2011 12th International Radar Symposium (IRS), Leipzig, 477–484. 2011.

[2] S. Heuel, and H. Rohling, “Pedestrian classification in automotive radar systems,” in 2012 13th International Radar Symposium, Warsaw, 39–44, 2012.

[3] S. Lee, Y.-J. Yoon, J.-E. Lee, and S.-C. Kim, “Human–vehicle classification using feature-based SVM in 77-GHz automotive FMCW radar”, IET Radar Sonar Navig. 11, 1589–1596, 2017, doi: 10.1049/iet-rsn.2017.0126.

[4] M. Ulrich, C. Gläser and F. Timm, "DeepReflecs: Deep Learning for Automotive Object Classification with Radar Reflections," 2021 IEEE Radar Conference (RadarConf21), Atlanta, GA, USA, 2021, pp. 1-6.

[5] B. Vogginger et al, “Automotive radar processing with spiking neural networks: concepts and challenges Frontiers in Neuroscience,” 2022.

[6] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, Bernhard Sick, ”Radar-based Road User Classification and Novelty Detection with Recurrent Neural Network Ensembles”, May 2019.

[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique”, Journal of Artificial Intelligence Research, Volume 16, Issue 1, Pages 321 – 357, 2002