applied sciences

MDPI

Article

Pedestrian and Multi-Class Vehicle Classification in Radar Systems Using Rulex Software on the Raspberry Pi

Ali Walid Daher [1,2,3,*], Ali Rizik [1,2], Andrea Randazzo [1], Emanuele Tavanti [1], Hussein Chible [2], Marco Muselli [3] and Daniele D. Caviglia [1]

1 Department of Electrical, Electronic and Telecommunications Engineering and Naval Architecture, University of Genova, 16145 Genoa, Italy; ali.rizik@edu.unige.it (A.R.); andrea.randazzo@unige.it (A.R.); emanuele.tavanti@edu.unige.it (E.T.); daniele.caviglia@unige.it (D.D.C.)

2 MECRL Laboratory, Ph.D. School for Sciences and Technology, Lebanese University, Beirut 6573/14, Lebanon; hchible@ul.edu.lb

3 Consiglio Nazionale Delle Ricerche, Institute of Electronics, Computer and Telecommunication Engineering, 16149 Genoa, Italy; marco.muselli@ieiit.cnr.it

* Correspondence: alidengineer@live.com

Received: 17 November 2020; Accepted: 17 December 2020; Published: 20 December 2020

Abstract: Nowadays, cities can be perceived as increasingly dangerous places. Usually, CCTV is one of the main technologies used in a modern security system. However, poor light situations or bad weather conditions (rain, fog, etc.) limit the detection capabilities of image-based systems. Microwave radar detection systems can be an answer to this limitation and take advantage of the results obtained by low-cost technologies for the automotive market. Transportation by car may be dangerous, and every year car accidents lead to the fatalities of many individuals. Humans require automated assistance when driving through detecting and correctly classifying approaching vehicles and, more importantly, pedestrians. In this paper, we present the application of machine learning to data collected by a 24 GHz short-range radar for urban classification. The training and testing take place on a Raspberry Pi as an edge computing node operating in a client/server arrangement. The software of choice is Rulex, a high-performance machine learning package controlled through a remote interface. Forecasts with a varying number of classes were performed with one, two, or three classes for vehicles and one for humans. Furthermore, we applied a single forecast for all four classes, as well as cascading forecasts in a tree-like structure while varying algorithms, cascading the block order, setting class weights, and varying the data splitting ratio for each forecast to improve prediction accuracy. In the experiments carried out for the validation of the presented approach, an accuracy of up to 100% for human classification and 96.67% for vehicles, in general, was obtained. Vehicle sub-classes were predicted with 90.63% accuracy for motorcycles and 77.34% accuracy for both cars and trucks.

Keywords: internet of things; edge computing; machine learning; cascading; radar

1. Introduction

As cities are getting smarter, and as the spread of intelligent surveillance technologies is gaining popularity within developed countries, making urban transport secure and efficient plays a key role in the safety of individuals as well as in affecting traffic flow, which, in turn, may negatively impact businesses within a city [1].

Unlike video cameras, the operation of short-range microwave radars is not much affected by the presence of adverse weather conditions. This fact makes them ideal, in addition to classical

closed-circuit television (CCTV) systems [2], for operating round-the-clock automatic surveillance in an urban environment. Radars [3–5] can be used for vehicle and pedestrian classification by relying on feature extraction from the range and Doppler profiles of each target. The data collected by radar measurements can be used as input to machine learning (ML) algorithms for classification. However, the hardware should be low cost and lightweight while providing good performance. Therefore, for this application, we have chosen a Raspberry Pi [6–8], a small portable edge computing device, which is a very effective platform for real-world scenarios as well as for educational and research purposes. Raspberry Pi is ideal for edge computing applications, where the node or embedded device possesses high processing capabilities and is required to have enough storage space to avoid cloud access. With the large number of wireless sensor nodes that are used over a wide range of applications, from wearable sensors to image processing and surveillance, along with the integration of artificial intelligence and machine learning in their decision making, they are required to be as smart as possible to avoid cloud access and reduce network traffic. Urban classification may not have cloud access and requires low latency, so it is the ideal example where both training and testing need to be applied on an edge computing node.

The main goal of the present work is the porting of the state-of-the-art machine learning package Rulex [9] onto the Raspberry Pi computational platform. The dependencies were compiled to produce output binary files that are compatible with the target platforms, the first being the Windows 32 Bit client and the second being the Raspberry Pi server, which is where the Rulex engine runs.

The remainder of this paper is organized as follows: Section 2 presents a review on pedestrian–vehicle classification, Section 3 introduces the porting of Rulex on the Raspberry Pi, and Section 4 describes the adopted machine learning architecture. Further, forecast results obtained with the present implementation are presented in Section 5. Finally, we draw some conclusions in Section 6.

2. Pedestrian–Vehicle Classification Review

Radar systems can be used for detecting and classifying different targets, such as pedestrians and vehicles [3–5]. Indeed, such systems produce, through a proper antenna, an electromagnetic wave that propagates to the objects eventually located in the inspected scenario. The targets interact with the impinging radiation employing the well-known scattering mechanism, generating a scattered field that partly returns to the radar receiver. Specifically, the reflected waves contain information about the characteristics of the objects that generated them.

The setup considered in the present paper includes a “Distance2Go” radar demo board developed by Infineon technologies, which is able to produce range-Doppler maps by performing a double fast Fourier transform (FFT) on the raw data measured using a frequency-modulated continuous wave (FMCW) scheme [5,10]. Such maps are characterized by a peak in correspondence to the frequency shift due to the propagation delay and to the Doppler effect (which is always present when dealing with moving targets). With proper processing, the main information related to the range and radial speed of the objects can then be easily obtained.

From these measurements, it is possible to derive features to be used for machine learning classification. Specifically, the machine learning features used for training are the extension of the range and velocity profiles, as well as the standard deviation, mean, and variance for the same variables, in addition to the radar cross-section and the estimated target velocity [5]. However, in vehicle classification, there is the problem of vehicles moving crosswise (i.e., along a direction perpendicular to the radar axis), which can be mistaken for pedestrians [11]. Indeed, longitudinal moving vehicles (i.e., traveling along the direction of the radar axis) have a large range profile and a point-shape velocity profile on the range-Doppler diagram. The opposite is true for pedestrians, due to multiple velocities caused by the movement of limbs. As for crosswise moving vehicles, their range profile is comparable to the range profile of pedestrians and their velocity profile may approach that of longitudinal moving vehicles. Consequently, we need to compute both transverse and longitudinal velocities, and with such additional features, we attempt to avoid misclassification. Another feature is the radar cross-section

(RCS), which is the equivalent scattering surface of the target seen by the radar and is related to the amount of power that is reflected by the object [12,13].

In the rest of this work, we will refer to data obtained with the system described in [5] that have been made available to us courtesy of the authors.

3. Porting Rulex to Raspberry Pi

Rulex is a machine learning software that supports various machine learning algorithms that can be easily applied in a user-friendly environment [9]. The Rulex Graphical User Interface (GUI) provides a means of importing training data and manipulating them before applying machine learning algorithms. The main proprietary algorithm for Rulex is the logic learning machine (LLM) [14,15] which implements explainable AI. LLM was the main algorithm used for most of the classifications, where a tree-based structure, which combines vehicle classes to achieve more accurate results, was adopted. Then, these new combined classes are split recursively until all vehicles have been classified in their respective sub-classes.

The Raspberry Pi is a credit card-sized personal computer, which can perform application-specific tasks as well as performing general-purpose everyday computing, where it can connect to most personal computer (PC) input/output hardware. The Raspberry Pi has multiple digital input/output pins which can be used in embedded system applications such as motor control, serial communications, liquid-crystal display (LCD), and interfacing with a practically infinite variety of sensors.

Nowadays, IoT devices are becoming more intelligent because they support artificial intelligence software and algorithms, so in this work, we have deployed Rulex to operate on the Raspberry Pi which is one of the most popular IoT hardware platforms. In order to port a software package from one platform to another, all of its internal and external dependencies should be compiled on the target platform. After compilation with a specific tool, binaries or executables are generated, which are a formatted version of the code to be linked to succeeding layers of the source code. Furthermore, before porting software from 64 Bits to 32 Bits, all of its dependencies should be compiled in 32 Bits. Visual Studio may be used to compile the libraries and code when porting to Windows 32 Bits. However, when porting to Linux, we used CMake [16,17], which is a cross-platform application for generating executables or libraries.

Rulex external libraries were ported to 32 Bits as the first step before compiling the entire code. We ported the source code on Windows 32 Bits which is the interface and Raspbian 32 Bits which is where the engine runs. During porting, one of the issues we faced was the incompatibility of some datatypes with 32-Bit hardware and software, so they were either changed or cast into a compatible datatype. Another issue is the inability to generate a larger number of threads, so we developed two software tests, where one was written in Python and the other in C/C++ as an attempt to find out what was the maximum number of threads that could be generated. Moreover, the source code was modified accordingly to optimize the maximum number of threads.

After compiling locally on Windows 32 Bits, we proceeded to remotely compile the source code. Rulex was also debugged remotely and made compatible with both operating systems by using various macros and correctly setting variable types.

4. Classification Architecture

4.1. Classification Methodology

In urban classification, there are usually four classes: a pedestrian class and three vehicle classes. We can go about running forecasts by relying on multiple algorithms in one overall simulation. Rulex possesses various algorithms to choose from, such as neural networks (NN), k-nearest neighbor (KNN), decision trees, support vector machines, and logic learning machine (LLM), all of which can be used for classification. However, since the adopted methodology applies multiple ML algorithms in a cascaded setup and tests multiple arrangements, which can have a large number of forecasts,

we included just NN and LLM. This was adopted since NN is a widely used ML algorithm and also because LLM is the commercial algorithm of Rulex. Furthermore, for NN, a multi-layer perceptron (MLP) arrangement was applied using the back propagation algorithm [18]. However, since the vehicle class can be split into 3 sub-classes, namely, cars, trucks, and motorcycles, and since there is no need to differentiate between them in real-world scenarios, we took advantage of this fact by applying a sub-class-based tree structure, where the classes are nodes and the algorithms are branches. The forecasts of the tree-based method can lead to improved validation results by setting different weights in each forecast. Therefore, we can choose a convenient split ratio between training and testing data, separately for each forecast, where the split method used is holdout validation. It is also possible to use different classification algorithms in each forecast on each branch. This is useful in case the adopted algorithm is not generating the expected results for the dataset at hand. Figure 1 presents such a tree structure, where the leaves are the final classification outputs.

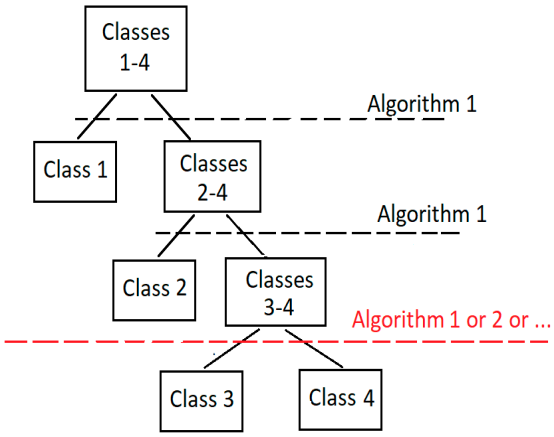

Figure 1. Sub-class-based tree structure.

As presented, different types of algorithms were applied to the final branches of the tree since they possess the lowest success rate. Therefore, the prediction accuracy is optimized by varying weights, data splitting, and, more critically, the algorithm used for classification. The further we go down the tree, the harder it gets to differentiate between classes.

4.2. System Setup

The machine learning system used consists of the Rulex Engine running on the Raspberry Pi as an application server, which is where forecasts are applied. This engine is accessed through a graphical interface running on a Windows client, while a PostgreSQL server is used as the common storage point between both nodes.

The data acquisition system has been described in [5]. It is a standalone system dedicated to feature extraction developed and operated separately concerning the client/server Raspberry Pi arrangement used to run Rulex for target classification. It consists of a Distance2Go radar board developed by Infineon technologies [19], a Raspberry Pi 3 B+, and a video camera oriented in the same direction as the radar beam. The system supply is provided by a 5 V, 10 Ah power bank that is sufficient for operating the whole system for several hours: for this work, it was installed on an internal road of the DITEN department of the University of Genoa. The system collects the raw data from the radar board through a Universal Serial Bus (USB) connection and extracts the features which are then forwarded to the processing chain of the Rulex software for classification.

5. Results and Discussion

The dataset collected by the data acquisition system described consists of 120 rows equally distributed into 4 classes with 30 patterns for every class. The features consist of the mean, variance,

and standard deviation of the range and Doppler profiles, along with their reflectivity and the estimated velocity of the target.

In order to maximize forecast accuracy, we have applied multiple tree-based sub-class arrangements to simulate using Rulex as a client/server setup [20].

As stated earlier, multiple cascaded simulations were applied with a varying number of classes as well as a cascaded order. Thus, a summary of all the applied forecasts is presented in Figure 2 and described in detail in this section. In Figure 2, the red labels stand for the cases and machine learning algorithms used in that particular simulation, and the green labels represent classes that will be split into sub-classes in the upcoming simulation.

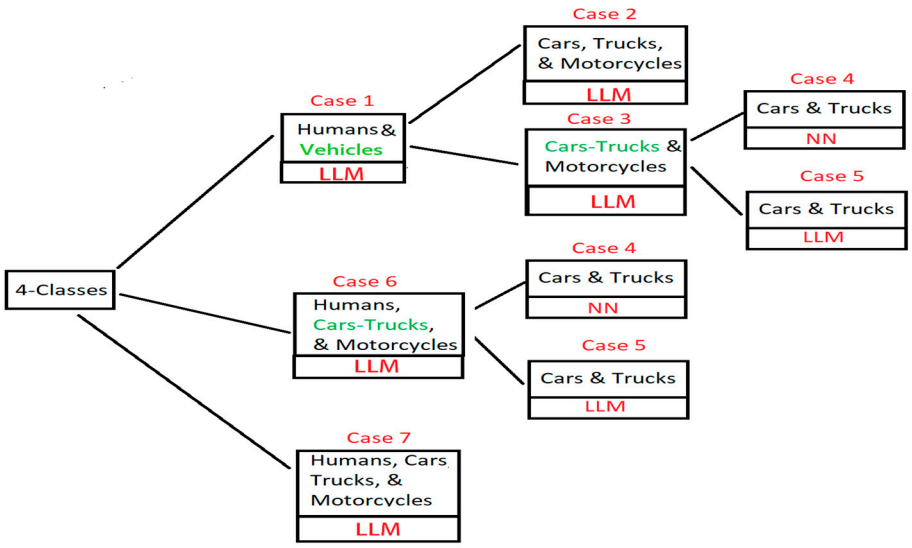

Figure 2. Cases 1–7 featuring all applied forecasts presented in this article.

In Case 1, we split the data into two classes: humans, and vehicles where LLM was used. The accuracy is shown in Table 1.

Table 1. Humans and vehicles training and testing prediction accuracy.

Class	Training	Testing
Humans	100%	100%
Vehicles	95%	100%

In Case 1, the machine learning algorithm used is LLM for classification. However, in Case 2, we only consider vehicle sub-classes. The simulation was applied using LLM where the prediction accuracies are found in Table 2.

Table 2. Vehicles, in 3 classes, training and testing prediction accuracy.

Class	Training	Testing
Cars	96%	75%
Motorcycles	94%	91%
Trucks	100%	72%

In Case 3, we apply a forecast using LLM for vehicle classes by splitting the data into two classes as shown in Table 3. In Cases 4 and 5, the cars/trucks class has been split into two sub-classes, cars and

trucks. In Case 4, we use neural networks, whereas LLM has been used in Case 5. The results of these can be found in Tables 4 and 5, respectively.

Table 3. Cars/trucks and motorcycles training and testing prediction accuracy.

Class	Training	Testing
Motorcycles	100%	91%
Cars/Trucks	98%	100%

Table 4. Cars and trucks with neural networks training and testing prediction accuracy.

Class	Training	Testing
Cars	95%	87%
Trucks	100%	70%

Table 5. Cars and trucks with LLM training and testing prediction accuracy.

Class	Training	Testing
Cars	95%	100%
Trucks	100%	66%

Cases 1 to 5 were processed separately to get a glimpse of how LLM would perform with this given dataset. From the outputs generated in Tables 1–5, we can estimate the overall prediction accuracy for a cascaded setup. Furthermore, it should be noted that misclassified records in preceding forecasts will be treated as correctly classified in upcoming forecasts, which leads to the overall accuracy of the cascaded system being incorrectly estimated. The preceding forecasts were all performed with a 70%/30% split for training and testing data, respectively, and with all weights being set to unity.

Furthermore, we can apply multiple cascaded setups which are based on the previous forecasts. If we cascade Cases 1 and 2, the projected output is presented in Table 6. In the case where we cascade Cases 1, 3, and 4, the projected output is provided in Table 7.

Table 6. Cases 1 and then 2 prediction accuracy.

Class	Forecast
Humans	100%
Motorcycles	91%
Cars	75%
Trucks	72%

Table 7. Cases 1, 3, and then 4 prediction accuracy.

Class	Forecast
Humans	100%
Motorcycles	91%
Cars	87%
Trucks	70%

If we cascade Cases 1, 3, and 5, we get the results shown in Table 8. Other variations of initializing the cascaded system with LLM can be found in Table 9, which is Case 6, where one class for humans along with two classes for vehicles is taken.

Table 8. Cases 1, 3, and then 5 prediction accuracy.

Class	Forecast
Humans	100%
Motorcycles	91%
Cars	100%
Trucks	66%

Table 9. Humans and vehicles, in 2 classes, training and testing prediction accuracy.

Class	Training	Testing
Cars/Trucks	97%	94%
Motorcycles	94%	100%
Humans	94%	69%

Finally, a single forecast for all four classes which is applied using LLM is presented in Table 10, namely, Case 7, which consists of forecasting all classes in a single block. With the variation added in Tables 9 and 10, we can apply two additional combinations to cascade. Thus, we can cascade Case 6 with Case 4 which employs neural networks or we can cascade it with Case 5 which uses LLM. These last two combinations include a situation where the previous prediction was not 100% accurate, so we need to take that into account when theoretically estimating the overall accuracy. When combining Case 6 with Case 4, the cars/trucks class has an accuracy of 94%, so, naturally, the neural network predictions in Case 4 will be multiplied by 0.94. The same can be said for Case 5, where the cars and trucks classes' success rates are multiplied by the same factor. Tables 11 and 12 provide the projected output forecast accuracy for the last two scenarios.

Table 10. Default LLM forecast training and testing prediction accuracy.

Class	Training	Testing
Humans	100%	73%
Cars	100%	100%
Motorcycles	84%	91%
Trucks	90%	72%

Table 11. Cases 6 and then 4 prediction accuracy.

Class	Forecast
Humans	100%
Motorcycles	69%
Cars	$0.94 \times 87 = 82\%$
Trucks	$0.94 \times 70 = 66\%$

Table 12. Cases 6 and then 5 prediction accuracy.

Class	Forecast
Humans	100%
Motorcycles	69%
Cars	$0.94 \times 100 = 94\%$
Trucks	$0.94 \times 66 = 62\%$

All of the preceding simulations only provide an estimation of actual results when cascading multiple engines. This is due to not considering the false-positive cases of the forecast. We took into consideration the entire dataset for each algorithm block and ignored some of the rows which were correctly classified in the abandoned class. In case two algorithms are cascaded, the first block should be followed by a Rulex data management block which will filter out the true and false positives in the abandoned class and remove them from the table. However, we still have to multiply the proceeding blocks with their parent class's success rate to calculate the overall accuracy.

For Case 1, we split the data 70/30, with 70% used for training and 30% being used for testing. The same was applied for Case 3. However, for Case 4, with the reduced number of rows, the data were split 65/35, with 65% used for training and the remaining 35% being used for testing. The main reason for changing the split ratio in Case 4 is due to the fact the prediction is applied to half of the dataset, and we found that increasing the size of the test set can lead to higher accuracy for the given data.

As for weights, the only way to set them and optimize results is by trial and error and intuition. There is no universal method to select weights accordingly. The unity gain in the Case 1 block should already provide very good results, so there is no need to change the weights. With a unity gain, in Case 3, the cars/trucks class, which will be used in the proceeding block, should be accurate while keeping the motorcycles class forecast precise enough. A gain of 1.5 was chosen for the cars/trucks field and 1.0 for motorcycles. As for the final block, which is Case 4, both trucks and cars classes, which originate in Case 3, have an equal true positive rate of 80% in testing. Therefore, weights are left at unity.

Table 13 represents the accuracy for training and testing of Cases 1, 3, and 5, respectively, and as predicted using Rulex. All the forecasts present good results for testing. Humans were detected with a rate of 100% and vehicles overall at a rate of 96.67%. In Case 3, which is block 2, the cars/trucks class has a true positive rate of 93.75% and motorcycles at 90%. As for the cars and trucks block, which is Case 5, the success rate is 80% for both trucks and cars. Tables 14 and 15 present the overall output true and false positive rates for the chosen All-LLM forecast. Humans are detected without any errors for the test dataset. The overall forecasts of the motorcycles, cars, and trucks have been calculated based on the preceding forecasts to become 90.63% for motorcycles and 77.34 for both cars and trucks.

Table 13. Training and testing accuracies for Cases 1, 3, and then 5 as predicted using Rulex before computing the actual accuracies.

Case	Training		Testing	
Case 1	Humans	Vehicles	Humans	Vehicles
	100%	95.82%	100%	96.67%
Case 3	Cars/Trucks	Motorcycles	Cars/Trucks	Motorcycles
	100%	100%	100%	93.75%
Case 5	Cars	Trucks	Cars	Trucks
	95%	100%	80%	80%

Table 14. Main forecast.

Class	Forecast
Humans	100%
Vehicles	96.67%

Table 15. Vehicles forecast.

Class	Forecast for Vehicles
Motorcycles	$0.9667 \times 93.75 = 90.63\%$
Cars	$0.9667 \times 80 = 77.336\%$
Trucks	$0.9667 \times 80 = 77.336\%$

Furthermore, the overall prediction of vehicles is 96.67%, which can be useful in practice.

6. Conclusions

The machine learning software Rulex has been ported to the Raspberry Pi in a client/server setup for edge computing applications. The device was used to make forecasts on a pedestrian and vehicle classification dataset for urban security applications. Multiple forecasts were cascaded in a tree-like structure while tuning the parameters of every forecast. Classes were split into sub-classes and single process simulations were applied, where we estimated the overall accuracy for various cascaded setups. After exhausting all the possible arrangements, the setup with the best projected output was simulated in a cascaded configuration, which provides an improved prediction outcome. This approach achieves higher accuracy over the classical approach of applying a single forecast for all classes. Further, combining classes into a parent class can be useful in practice, such is the case with the vehicles parent class. However, this approach is exhaustive and time-consuming and may require setting parameters for different forecasts and various ML algorithms. Moreover, the tree-based method for improving machine learning forecasts can be used in various configurations. After applying the proposed method, humans were classified with an accuracy of 100% and vehicles with an accuracy of 96.67%. The final vehicles sub-classes forecast accuracies are 90.63% for motorcycles and 77.34% for the cars and trucks classes.

Author Contributions: A.W.D. developed and applied the main machine learning method. A.W.D. Wrote the main draft. A.W.D. and M.M. ported Rulex onto the Raspberry Pi. A.R. (Ali Rizik) collected the data. A.W.D., A.R. (Ali Rizik), A.R. (Andrea Randazzo), E.T., H.C., M.M. and D.D.C. discussed the results and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Carter, B. Making Cities Smarter, Urban Population Is Posing New Set of Complex Security Challenges. Available online: https://www.securitytoday.com/articles/ (accessed on 9 April 2020).
2. Clive, N.; McCahill, M.; Wood, D. The growth of CCTV: A global perspective on the international diffusion of video surveillance in publicly accessible space. *Surveill. Soc.* **2002**, *2*, 110–135.
3. Rohling, H.; Heuel, S.; Ritter, H. Pedestrian Detection Procedure Integrated into an 24 GHz Automotive Radar. In Proceedings of the 2010 IEEE Radar Conference, Washington, DC, USA, 10–14 May 2010; pp. 1229–1232.
4. Prophet, R.; Hoffmann, M.; Ossowska, A.; Malik, W.; Sturm, C.; Vossiek, M. Pedestrian Classification for 79 GHz Automotive Radar Systems. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1265–1270.
5. Rizik, A.; Randazzo, A.; Vio, R.; Delucchi, A.; Chible, H.; Caviglia, D.D. Feature Extraction for Human-Vehicle Classification in FMCW Radar. In Proceedings of the 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 27–29 November 2019; pp. 131–132.
6. Xingzhou, Z.; Wang, Y.; Shi, W. pCAMP: Performance Comparison of Machine Learning Packages on the Edges. In Proceedings of the USENIX Workshop on Hot Topics in Edge Computing (HotEdge '18), Boston, MA, USA, 10 July 2018.
7. Vladimir, V.; Maksimović, M. Raspberry Pi as a Wireless Sensor Node: Performances and Constraints. In Proceedings of the 37th IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 26–30 May 2014; p. 10131018.
8. Maksimović, M.; Vujović, V.; Davidović, N.; Milošević, V.; Perišić, B. Raspberry Pi as Internet of Things Hardware: Performances and Constraints. In Proceedings of the 1st International Conference on Electrical, Electronic and Computing Engineering IcETRAN 2014, Vrnjačka Banja, Serbia, 2–5 June 2014.
9. Marco, M. Extracting knowledge from biomedical data through Logic Learning Machines and Rulex. *EMBnet* **2012**, *18*, 56–58.

10. Wojtkiewicz, J.; Misiurewicz, M.N.; Jedrze-jewski, K.; Kulpa, K. Two-Dimensional Signal Processing in FMCW Radars. In Proceedings of the XX National Conference on Circuit Theory and Electronic Networks, KKTOiUE'97, Kołobrzeg, Poland, 21–24 October 1997; Volume 2, pp. 475–480.
11. Steffen, H.; Rohling, H. Pedestrian Classification in Automotive Radar Systems. In Proceedings of the 13th IEEE International Radar Symposium, Warsaw, Poland, 23–25 May 2012; pp. 39–44.
12. Skolnik, M.I. *Introduction to Radar Systems*; McGraw-Hill: New York, NY, USA, 2001.
13. Liaqat, S.; Khan, S.A.; Ihsan, M.B.; Asghar, S.Z.; Ejaz, A.; Bhatti, A.I. Automatic Recognition of Ground Radar Targets Based on Target RCS and Short Time Spectrum Variance. In Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 15–18 April 2011; pp. 164–167.
14. Marco, M. Switching neural networks: A new connectionist model for classification. In *Neural Nets*; WIRN 2005, NAIS 2005. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3931, pp. 23–30.
15. Muselli, M.; Ferrari, E. Coupling Logical Analysis of Data and Shadow Clustering for Partially Defined Positive Boolean Function Reconstruction. In Proceedings of the IEEE Transactions on Knowledge and Data Engineering, Washington, DC, USA, 23 January 2009; pp. 37–50.
16. Dominique, F.; Orlarey, Y.; Letz, S. Building Faust with CMake. In Proceedings of the 1st International Faust Conference (IFC-18), Mainz, Germany, 18 June 2018.
17. Clemencic, M.; Mato, P. A CMake-based build and configuration framework. *J. Phys. Conf. Ser.* **2012**, *396*, 052021. [CrossRef]
18. Rumelhart, E.D.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; No. ICS-8506; University of California San Diego La Jolla Department for Cognitive Science: Fort Belvo, CA, USA, 1985.
19. Demo Distance2Go. Available online: https://www.infineon.com/cms/en/product/evaluation-boards/demo-distance2go/ (accessed on 22 August 2017).
20. Kemal, H.; Konjicija, S.; Subasi, A. A Low Energy APRS-IS Client-Server Infrastructure Implementation Using Raspberry Pi. In Proceedings of the 22nd IEEE Telecommunications Forum, Telfor (TELFOR), Belgrade, Serbia, 25–27 November 2014; pp. 296–299.