A testing framework for DNN-based radar applications

Mojdeh Golagha

mojdeh.golagha@infineon.com

Abstract: Nowadays, radar systems play a vital role in assisted driving for adaptive cruise control and obstacle avoidance. Practitioners apply neural network architectures to radar data for object detection and classification. Deep neural networks (DNN) can automatically extract the features from the radar range-doppler signatures such that they generally achieve high classification accuracy. However, one concern that is often cited when designing DNN-based radar applications is their resilience against noise, perturbation, distribution shift, and adversarial attacks. So, there is a need to investigate their trustworthiness before using them in the real world in safety-critical domains. In this paper, we offer a fuzz test generation framework to evaluate the performance of DNNs. Our framework automatically generates test cases valuable for improving model robustness by recognizing more diverse error-triggering inputs. Fuzzing systems work based on mutating input data while preserving the semantic labels.

Radar is an electromagnetic system that uses radio waves to detect and localize target objects which reflect a signal. The time taken by the signals to return from the obstacles to the device is used to determine the angle, range, and velocity of the objects. Radars are being extensively used to make smart homes, factories, healthcare, aerospace, defense, and so on.

Radar plays a crucial role in automated driving today. In order to become a leader in autonomous vehicle manufacturing, all automobile manufacturers have entered the field, and they are competing to develop the most advanced autonomous vehicles. In order to achieve this goal, automobile companies focus their research and development efforts on sensor technologies as autonomous vehicles rely heavily on sensors for navigation. For a seamless ride, most autonomous vehicles use a combination of cameras, lidar, and radar for imaging, detection, ranging, tracking, and sensing location. Advanced driving assistance systems (ADAS) in autonomous vehicles utilize sensors to provide more precision and power.

Compared to other types of sensors used in autonomous vehicles, radar is particularly reliable in low visibility conditions, such as cloudy weather, snow, rain, and fog. Autonomous vehicles use radar that operates in the frequencies of 24, 74, 77, and 79 GHz. These frequencies correspond to short-range radars (SRR), which are used for blind-spot monitoring, lane-keeping assistance, and parking assistance; medium-range radars (MRR) that are used for obstacle detection in the range of 100-150 meters; and long-range radars (LRR) that are used for automatic distance controlling and brake assistance. Typical automotive radar applications include adaptive cruise control (ACC), blind-spot detection (BSD), line change assistant, and so on.

AI algorithms can be used to interpret radar measurements in an advanced way. Neural networks are used to automatically classify radar targets, which allows for a deeper understanding of radar signals and images. For instance, in [PHV$^+$18], Prophet et al. have developed a classification system for pedestrians using a 79 GHz automotive radar sensor. The ability to classify objects is used to distinguish vulnerable road users from other objects, such as vehicles. They use a 79 GHz chirp sequence to create a range-Doppler-Matrix. This matrix indicates the location and velocity of all the targets in the area.

In another application [MGZV18], Martínez García et al. propose a convolutional neural network (CNN) for classifying radar images in order to detect vacant parking spaces with a 77-GHz imaging radar.

The emerging use of AI-based radar applications increases the need for systematic testing and thorough evaluation of such safety-critical systems as other technical systems. Any risks have to be addressed by employing suitable measures. The same hidden faults that cause incorrect behavior in traditional software could also cause major accidents and losses in AI applications.

In this paper, we prose a framework for detecting potential defects of deep neural networks (DNN) designed for radar applications. Our framework is designed based on the metamorphic testing strategy [CCY20] and is guided through some coverage criteria. Utilizing this framework, users can detect corner cases and generate tests automatically.

1 Challenges of testing DNN-based applications

DNNs have made remarkable progress in image and speech recognition in the past few years. Self-driving cars are increasingly using these technologies to do advanced autonomous tasks. To ensure the safety and security of such systems, practitioners need to test and evaluate them using realistic scenarios. The standard way of measuring the performance of such algorithms is calculating metrics such as accuracy, F-score, an AUC [MSTS19]. However, these metrics can be insufficient because they are calculated on mostly hand-picked and randomly chosen test datasets. The test dataset might not be a good representative of the real-world data or the training data distribution [MSTS19].

The quality of the test dataset is an important factor in the trustworthiness of accuracy results. If the test dataset does not cover the distribution of training data or is not diverse enough to cover real-world scenarios, the accuracy results cannot be trusted. In addition test dataset must be a combination of hard and easy samples to ensure robustness[WXK$^+$21]. Therefore, there is a need for a systematic way of test generation to find corner cases and explore untested areas [MSTS19]. To this end, we propose a coverage-guided fuzz test generation technique.

2 Test generation framework

Figure 1 indicates the main components of our testing framework. The framework is based on DeepHunter [XMJX$^+$19], but has been modified to adapt radar data type and applicati-

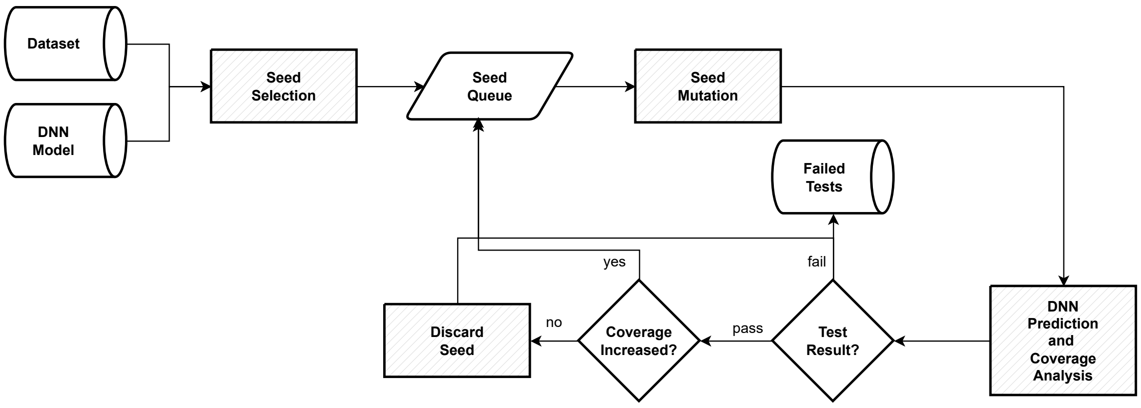

Abbildung 1: Test Generation Framework

ons. The overall structure of the fuzzing procedure is the following: the fuzzer starts with an initial seed queue containing at least one set of valid inputs for the DNN under test. Given this seed queue, until instructed to stop, the fuzzer selects seeds from the queue. Given the seed as input, the mutator component performs semantic preserving transformations to that input. Finally, the mutated inputs are fed to the DNN. While executing, the coverage at different levels of granularity is extracted. Failing tests are added to the failing tests pool and are added to the queue again for further mutations. Passing tests that exercise new coverage are also added to the queue again. But, passing tests without coverage increase would be discarded to avoid wasting more time on them. This process continues until some ending criteria are satisfied. The ending criteria are defined by the user.

In the following, we describe different components in more detail.

2.1 Seed selection:

To start fuzzing, we need initial seeds on which DNN has high performance. However, initial seeds can be selected in two ways: a) data points classified correctly with a confidence score higher than a threshold, or b) the entire test dataset. The next step is making a queue of these initial seeds. The fuzzer selects seeds from this queue to mutate.

2.2 Seed mutation:

The selected seeds from the queue are sequentially mutated using defined mutation criteria. Depending on the data type, different transformers should be defined. Examples of image transformers include blur, shear, rotate, auto-contrast, noise, and so on. There are more advanced transformers such as Augmix [HMC+19] and Augmax [WXK+21].

We restrict the mutant data to have Structural Similarity Index Measure [BBS+22]

$$(SSIM) > 0.5$$

to preserve the semantics of the image. SSIM is used since it is a perceptual metric that reflects the image quality degradation of similar-looking images.

2.3 Coverage criteria:

After mutating seeds, they are fed to the DNN for prediction. Meanwhile, the coverage is computed. Coverage criteria used in our framework are based on [YPW21] and [XMJX$^+$19].

- **Neuron Coverage (NC)**: NC can be compared with the code coverage in traditional software testing. NC is the ratio of activated neurons in the DNN for corresponding test input. Here, we need to define a threshold value. So, the neuron is considered to be activated when the output value for that neuron is above the threshold. The default value is 0.75.
- **K-Multi Section Neuron coverage (k-MNC)**: The neuron's training range is divided into k sections. A value section is considered to be covered when the output value of the neuron falls into this region using the test input. k-MNC is a measure of covered sections of all trainable neurons of DNN. A diverse set of inputs are required to maximize the k-MNC coverage criteria. The default value for k is 1000.
- **Neuron Boundary Coverage (NBC)**: NBC is a ratio of corner-case regions covered by given input. It is calculated by using both upper and lower boundary values. The default value is 10.
- **NeuraL Coverage (NLC)**: As opposed to treating each neuron as a separate computational unit, NLC considers each DNN layer to be the basic computational unit and captures four important features of neuron output distributions, namely divergence, correlation, shape, and density.

2.4 Output

The outputs of the fuzzer are – a) coverage information, b) identified corner cases and a new c) test dataset. The coverage information could help us to evaluate the performance of the trained DNN under various coverage mutation criteria. The identified corner cases are hints to detect potential defects of a trained DNN. By analyzing the crash data, we can identify useful transformations for which DNN might have biased performance. This information could be very useful to re-iterate the training pipeline.

3 Radar specific settings

As mentioned before, radars allow us to estimate the range, velocity, and angles of targets. It is typically necessary to use a preprocessing chain to extract some of these parameters from the sampled intermediate frequency signal, before feeding it into the neural network, in order to improve interpretability. Different such preprocessing chains are given in [SH20]. The input to the neural networks are Range-Doppler Images (RDI) of each radar's antenna [ZLW20] for object detection, or directly the time domain data [SSF20], which are then used implicitly to generate RDIs in the first network layer.

Since RDI images are different from usual images, the introduced transformation for images should not be applied to them. The mutation techniques should be meaningful for this type of data. Therefore, in our mutation component, we implemented data augmentation techniques such as time shift, frequency disturbance, and frequency shift introduced in [SLY21], [SWW20] as well as transformations using generative adversarial networks [AI19].

4 Conclusion

In this paper, we presented a testing framework for evaluating and improving the quality of AI-based radar applications. Our proposed framework is based on the metamorphic testing strategy. Different components of the framework have been customized to adapt radar-specific data types. Different radar data transformations have been identified and used for generating new tests. Identified corner cases and failing tests can be used to retrain the DNN. This leads to an improvement in the robustness of the algorithm.

Literatur

[AI19] Kim Youngwook Alnujaim Ibrahim. Augmentation of Doppler Radar Data Using Generative Adversarial Network for Human Motion Analysis. *Healthc Inform Res*, 25(4):344–349, 2019.

[BBS+22] Illya Bakurov, Marco Buzzelli, Raimondo Schettini, Mauro Castelli und Leonardo Vanneschi. Structural similarity index (SSIM) revisited: A data-driven approach. *Expert Systems with Applications*, 189:116087, 2022.

[CCY20] Tsong Yueh Chen, S. C. Cheung und Siu-Ming Yiu. Metamorphic Testing: A New Approach for Generating Next Test Cases. *CoRR*, abs/2002.12543, 2020.

[HMC+19] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer und Balaji Lakshminarayanan. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty, 2019.

[MGZV18] Javier Martínez García, Dominik Zoeke und Martin Vossiek. MIMO-FMCW Radar-Based Parking Monitoring Application With a Modified Convolutional Neural Network With Spatial Priors. *IEEE Access*, 6:41391–41398, 2018.

[MSTS19] Senthil Mani, Anush Sankaran, Srikanth Tamilselvam und Akshay Sethi. Coverage Testing of Deep Learning Models using Dataset Characterization. *CoRR*, abs/1911.07309, 2019.

[PHV$^+$18] Robert Prophet, Marcel Hoffmann, Martin Vossiek, Christian Sturm, Alicja Ossowska, Waqas Malik und Urs Lübbert. Pedestrian Classification with a 79 GHz Automotive Radar Sensor. In *2018 19th International Radar Symposium (IRS)*, Seiten 1–6, 2018.

[SH20] A. Santra und S. Hazra. *Deep Learning Applications of Short-range Radars*. Artech House Radar series. Artech House, 2020.

[SLY21] Donghong She, Xin Lou und Wenbin Ye. RadarSpecAugment: A Simple Data Augmentation Method for Radar-Based Human Activity Recognition. *IEEE Sensors Letters*, 5(4):1–4, 2021.

[SSF20] Michael Stephan, Avik Santra und Georg Fischer. Human Target Detection and Localization with Radars Using Deep Learning. 2020.

[SWW20] Marcel Sheeny, Andrew Wallace und Sen Wang. RADIO: Parameterized Generative Radar Data Augmentation for Small Datasets. *Applied Sciences*, 10, 06 2020.

[WXK$^+$21] Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar und Zhangyang Wang. AugMax: Adversarial Composition of Random Augmentations for Robust Training. *CoRR*, abs/2110.13771, 2021.

[XMJX$^+$19] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin und Simon See. DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2019, Seite 146–157, New York, NY, USA, 2019. Association for Computing Machinery.

[YPW21] Yuanyuan Yuan, Qi Pang und Shuai Wang. You Can't See the Forest for Its Trees: Assessing Deep Neural Network Testing via NeuraL Coverage, 2021.

[ZLW20] Guoqiang Zhang, Haopeng Li und Fabian Wenger. Object Detection and 3d Estimation Via an FMCW Radar Using a Fully Convolutional Network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seiten 4487–4491, 2020.