# Environment-aware Testing for DNN-based Smart-home WiFi Sensing Systems

Naiyu Zheng[1,3,*], Ting Chen[2], Chuchu Dong[2], Yubo Yang[2], Yuanzhe Li[1], Yunxin Liu[1,4], Yuanchun Li[1,4,†]
[1] *Institute for AI Industry Research (AIR), Tsinghua University*
[2] *Midea Group Co., Ltd.*
[3] *Beijing University of Posts and Telecommunications*
[4] *Shanghai AI Laboratory*

***Abstract*—WiFi-based human activity recognition is a promising sensing application in smart homes due to the low cost, wide availability, and privacy preservation of WiFi devices. However, pushing WiFi sensing technology to industry-scale deployment is difficult due to its poor robustness against environment differences. How to systematically test such sensing system is crucial to improve its practicality, and is also challenging because the sensing performance is significantly influenced by the underlying physical environments. In this paper, we introduce the problem of testing environment-dependent sensing systems, including how to measure test coverage and how to effectively generate data to improve the coverage. We describe our initial attempts on examining test sufficiency with environment-neuron joint coverage and improving the coverage through targeted environment variations and signal transformations. Our experiments have demonstrated the higher effectiveness of using environment-neuron coverage to represent test sufficiency, as compared with using the conventional neuron coverage. Meanwhile, the coverage-guided sensing data generation can lead to higher accuracy of the sensing system under changing environments.**

***Index Terms*—Smart home, software testing, WiFi sensing, deep learning, environment dependency.**

## I. Introduction

Recent years have witnessed rapid progress of wireless communication technology and large-scale adoption of WiFi-based IoT devices. It is widely known today that the digital signals transmitted and received by these devices can be used to sense our states and activities [8], which enables many useful smart-home applications such as gesture recognition, sleep monitoring, fall detection, etc. Meanwhile, recent advances of artificial intelligence (AI) and deep learning (DL) make it much easier to develop sensing applications by training models with the wireless signals.

However, pushing WiFi sensing technology to commercialization still faces several important challenges, a major one of which is the robustness of the sensing system in diverse and dynamic smart-home environments. Specifically, when using wireless signals in smart homes for user activity recognition, the difference between homes and the change of furnitures may lead to significantly different sensor readings [20], [1]. It is difficult even impossible for sensing service providers to manually develop a model for every home, while the models developed in one environment (*e.g.* the developers' laboratory) are usually unable to achieve acceptable accuracy in other environments (*e.g.* the end-users' homes). How to systematically test the smart-home WiFi sensing system is crucial for understanding the limitations of a sensing system and improving it accordingly.

Unlike traditional software that can be tested with random inputs to achieve high coverage, the robustness of sensing systems can only be measured with realistic sensor signals. Generating fake and random sensing readings would lead to meaningless test results. Existing work has also explored various data generation techniques to test learning-based systems such as self-driving car and machine translation [10], [9]. However, these techniques can hardly be applied to test smart-home sensing systems, because the sensor signals are much less human-understandable as compared with images or natural language sentences.

To this end, we focus on the problem of robustness testing for smart-home sensing systems. The challenges of testing sensing system include two main aspects. First, it is unclear how to systematically evaluate the test coverage so that the coverage value can reflect the system performance in actual environments. Second, due to the special characteristics of sensing signals, it is difficult to effectively generate data to improve the coverage and therefore enhance the system robustness.

As an initial attempt to address the above problem, we propose to use environment-neuron joint coverage to measure test progress instead of conventional neuron coverage. Specifically, the test coverage is represented as a product of the neuron coverage on each environment property, where the weight is determined by the distribution of the property values in real world. Guided by this coverage metric, developers can make more informed decisions when determining new environments for data collection. Additionally, we introduce techniques to improve the test coverage through domain-specific signal transformation techniques, including window-shifting transformations and frequency-scaling transformations.

We evaluate our solution on a popular public dataset named Widar3.0 [20]. The results have shown that the environment-neuron joint coverage can more effectively describe the test progress since it can better reflect the reliability of sensing system in real deployment. Meanwhile, our coverage-guided

* Naiyu Zheng did this work while as an intern at Institute for AI Industry Research (AIR), Tsinghua University. †Correspondence goes to Yuanchun Li (liyuanchun@air.tsinghua.edu.cn)

data generation techniques can improve the test coverage by 19.13%-35.29% and improve the cross-environment sensing accuracy by 11.78%-16.70%.

The contributions of this work include:

1) We introduce the problem of testing smart-home sensing systems, which provides a new perspective of software testing where the software behavior is highly dependent on the underlying environments.
2) We introduce the environment-neuron joint coverage to measure test coverage for smart-home sensing systems, which is a more approperiate criterion than conventional neuron coverage.
3) We introduce a data generation technique based on coverage-guided new environment determination and sensor signal transformation, which helps improving test coverage and lead to more robust sensing systems.

## II. Background and Related Work

### A. Smart-home WiFi Sensing

Using WiFi signals for sensing has become a popular research direction for a long time [8]. Specifically, WiFi-based human activity sensing in smart homes is believed to be a killer application. The common practice of WiFi sensing is to obtain WiFi signals with two connected devices (a transmitter and a receiver), extract the Channel State Information (CSI) from the WiFi signals, transform it, and use deep learning (DL) to convert the information to high-level predictions.

It has been demonstrated that human states such as running and walking can be detected based on the amplitude information of CSI [22], [14], and the human movement can be obtained by analyzing the CSI value dynamics [12]. The Fresnel zone model [16] proved the sensing feasibility of decimeter-scale activity recognition by WiFi signals theoretically. The WiFi signals can be further processed to extract information that better depicts human activities. A typical example is the Doppler frequency shift (DFS) [4], [20], which is obtained by transforming CSI from time domain to frequency domain. The sensing system in our work is also based on the DFS information.

A major challenge of wireless sensing is the high environmental dependence of wireless signals, *i.e*. the sensor readings for the same activity in different environments may be significantly different. To address this problem, researchers have adopted various domain adaptation techniques to generalize sensing systems across different environments [1] or introduced domain-independent representations to better depict human activities [20].

Unfortunately, pushing WiFi-based deep sensing technology to commercialization is still difficult. One of the major obstacles is the lack of standard and effective testing methods in this field.

### B. Deep Learning Testing

Deep WiFi sensing system is a special type of software based on deep neural networks (DNNs), while how to test and improve the quality of DNN-based systems is a rapidly-developing research direction in the software engineering (SE) community. In recent years we have witnessed ongoing efforts in the SE community to measure and enhance the robustness of DL models through testing. The main research topics in this domain include (1) how to measure the test coverage for DL-based applications and (2) how to generate inputs to improve the test coverage.

The most widely-used test criterion for measuring test coverage of DL models is neuron coverage [9], which is represented as the ratio of activated neurons among all neurons in the model. Based on neuron coverage, researchers have further introduced more fine-grained metrics [6] such as K-multisection neuron coverage and neuron boundary coverage. Researchers have also discussed the effectiveness of neuron coverage in DNN testing [2], [15], [13].

To maximize test coverage, various techniques are developed to synthesize inputs to thoroughly test the DNNs [17], [10], [21]. Another line of work tries to probe the boundaries with test inputs and understand when the model could yield erroneous prediction [3], [18], [11], [5], [7]. This analysis can be done by assessing the models' uncertainty [18], [7], detecting out-of-distribution (OOD) data [3], [11], [19], or directly predicting the failure of the AI system [5].

These techniques are mostly designed for computer vision or natural language processing applications, while the unique environment-dependent characteristic of sensing applications is rarely discussed and addressed.

## III. Environment-aware Testing

### A. Problem Description

We first introduce the problem of smart-home sensing system testing. The goal of testing is to understand and enhance the reliability of the target sensing system across different environments of deployment. Specifically, the performance of the sensing system in one environment $e$ can be measured by its sensing accuracy $acc_e$ (*i.e*. the correctness of predicted human activities in the environment). The reliability across different realistic environments is represented as $acc_{E_{real}} = \mathbb{E}_{e \in E_{real}}\ acc_e$, where $E_{real}$ is the set of real-world sensor deployment environments (*e.g*. all end-user homes). However, measuring the sensing performance in each $e \in E_{real}$ is difficult even impossible in practice. Instead, we usually only have access to a small subset of test environments $E_{dev}$ that are controlled by the developers. Thus, we need an easy-to-obtain testing criterion to estimate the reliability of the target sensing system in the wild, and guide test input generation to achieve higher reliability.

Due to the huge environment dependency of WiFi sensing systems, the conventional neuron coverage may not be a satisfactory criterion, since it may lead to over-approximation of test adequacy. Specifically, the extensive sensor data in a single environment may produce high neuron coverage in the model, but it may not cover diverse data distributions in different environments. Therefore, a more domain-specific metric to measure test adequacy of sensing model is needed.

Meanwhile, how to maximize the test coverage with limited time and labor effort is also an important question in sensing system testing. Unlike most traditional software systems whose inputs are semantically understandable and can be manually constructed, the inputs of the sensing systems are sensor signals that can only be produced in certain environments. Thus, generating data for sensing systems requires wisely determining the testing environments and better utilization of the data in the limited environments.

The overview of our solution is shown in Figure 1. The main components of the solution include a new coverage criterion (environment-neuron joint coverage) and a coverage-guided input generation method. The details of the two components are described below.

### B. Environment-Neuron Joint Coverage

The purpose of environment-neuron joint coverage is to describe not only the test coverage of cases that may be encountered in a certain environment, but also the cases that may be found in different environments. In this way, the joint coverage can more precisely reflect the robustness of the smart-home sensing system in different environments.

To obtain the environment-neuron joint coverage, the sensing model $M$ should first be tested in several existing environments $E_{dev}$ that are controlled by the developer. Suppose the dataset in each environment $e \in E_{dev}$ is $D_e$, we use $CN_e$ to denote the neurons covered in environment $e$ by feeding $D_e$ to model $M$, *i.e.* $CN_e = covered\_neurons(M, D_e)$.

Next, for each environment $e \in E_{dev}$, we analyze its covered properties in real environments. Specifically, developers should understand the environment properties that may affect the data distribution of sensing signals. For example, the important environment properties in WiFi sensing include the location and orientation of the target user, the distance and number of obstacles between the transmitter and the receiver, the number of other moving objects in the room, etc. Suppose the distribution of all environment properties is $P = \{P_1, P_2, ..., P_k\}$, where $k$ is the number of properties and each $P_i$ is the distribution of the values of a certain property, *i.e.* $P_i(p)$ is the probability of the $i$-th property having the value $p$ in real environments. Based on the definition of properties, we can characterize each environment $e$ with the property values $p^e = \{p_1^e, p_2^e, ..., p_k^e\}$.

Combining the covered neurons and covered properties in each environment, we can obtain the covered neurons for each value $p_i$ of environment property $P_i$ as

$$covered\_neurons(p_i, E_{dev}) = \bigcup_{e \in E_{dev} \ and \ p_i^e = p_i} CN_e \quad (1)$$

Then the test coverage for each environment property can be calculated as

$$coverage(P_i, E_{dev}) = \sum_{p_i \in P_i.values} P_i(p_i) \frac{covered\_neurons(p_i, E_{dev})}{total\_neurons(M)} \quad (2)$$

Finally, the overall coverage can be obtained by

$$coverage(P, E_{dev}) = \prod_{P_i \in P} coverage(P_i, E_{dev}) \quad (3)$$

Such a coverage criterion can handle the diversity of real environments and reflect the test sufficiency of the sensing system in actual deployment.

### C. Coverage-guided Input Generation

Guided by the environment-neuron joint coverage, we can strategically generate sensing data to improve the coverage and consequently enhance the reliability of the sensing model in real deployment environments.

First, the coverage metric can guide the developers on the determination of new environments for data collection that can help achieve higher coverage more efficiently. Specifically, suppose the developers have the budget to create one new environment for data collection. Instead of arbitrarily choose an environment, an ideal choice is a new environment that can more easily lead to the improvement of overall test coverage, i.e. $e_{new} = argmax_{e'} coverage(P, E_{dev} \cup \{e'\})$. Directly solving this equation is infeasible because the covered neurons in new environments are unknown in advance, so we heuristically choose the environment with the most important (common in real environments) but less-covered environment property values.

Second, in each test environment $e \in E_{dev}$, developers can augment the sensor data to obtain higher coverage. Based on the insight that the prediction of human activity should remain consistent in a small duration of time or with slightly-different movement speed, we propose to generate meaningful sensing signals from existing signals with time-domain transformations. Specifically, an activity sample is collected at 1000Hz for about 3 seconds, and we can get a sequence of sensor signals $\{x_t, x_{t-d}, ..., x_{t-Nd}\}$ through fixed interval sampling, where $t$ is the end time of the activity and $d$ is the interval between successive sensor readings. Since the starting position and interval of sampling do not directly affect the overall features of the activity, We can further obtain more samples for the same activity as $\{x_{t'}, x_{t'-d'}, ..., x_{t'-Nd'}\}$, where $t' = t + \Delta t$ and $d' = \sigma d$ are slightly and randomly shifted end time and scaled sampling duration. In this way, the same activity segment can be used to generate multiple samples which greatly increases the diversity of data. In our experiment, we generate more than 10x data samples from existing data with this technique.

## IV. Evaluation

### A. Experiment Setup

We conduct experiments with a public dataset named Widar3.0 [20], which contains the WiFi sensing data samples corresponding different gestures obtained in multiple environments across different rooms. Each environment in a room is characterized by two properties, *i.e.* the location and orientation of the user. There are 25 combinations (5 locations and 5 orientations) in total in each room. We create three

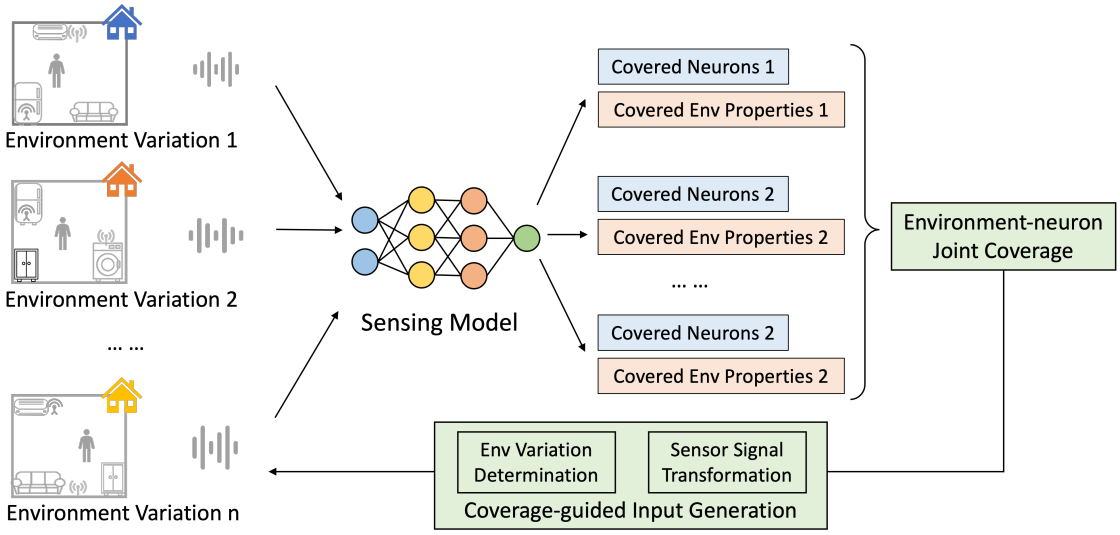


Fig. 1: The overview of our testing framework.

TABLE I: The correlation between different coverage metrics and actual performance. $E_{dev}$ represents the number of environments used by the developer to train and test the model. $Acc_{dev}$ and $Acc_{real}$ are the sensing accuracy achieved in the development environments and real environments. "Neuron Coverage" is the conventional coverage metric [9] and "Joint Coverage" is our proposed environment-neuron joint coverage.

| $E_{dev}$ | $Acc_{dev}$ | $Acc_{real}$ | Neuron Coverage | Joint Coverage |
|---|---|---|---|---|
| 3 envs | 97.62% | 48.45% | 98.24% | 15.41% |
| 6 envs | 94.05% | 49.54% | 98.63% | 62.26% |
| 9 envs | 95.24% | 59.82% | 98.63% | 77.83% |

environments with 3, 6, and 9 random location-orientation combinations in one room respectively as the development environment $E_{dev}$. All location-orientation combinations in another room are regarded as the real environments $E_{real}$. We use 90% random data samples in $E_{dev}$ to train the sensing model for 100 epochs to ensure convergence. The other 10% samples are used to test the model and measure test coverage. The actual performance of the sensing model is measured on the real-environment ($E_{real}$) samples. The performance metric is classification accuracy. Specifically, we selected the samples of six gestures (Slide, Swipe, Push&Pull, Clap, Draw-Z, and Draw-O) and examine how accurately the models can predict the gesture for each sensor data sample.

### B. Effectiveness of Environment-Neuron Joint Coverage

We first evaluate the effectiveness of our coverage metric in comparison with the conventional neuron coverage metric.

Table I shows the correlation between the accuracy in real environments $Acc_{real}$ and different metrics. First, the accuracy in test environments ($Acc_{dev}$) is high (all above 94%) in all environment settings, but it is not closely related to the actual performance ($Acc_{real}$). This is because of the data distribution difference between test environments and real environments.

The neuron coverage is also not a good indicator of the sensing performance in real world. Specifically, the neuron coverage values are almost equally high (above 98%) for different $Acc_{real}$ values. The main reason is that the sensing data is diverse while the sensing model is relatively small, making it easy to achieve a high neuron coverage.

Instead, the environment-neuron joint coverage proposed by us is much more correlated to the real-environment accuracy, where a higher joint coverage can constantly map to higher $Acc_{real}$. This demonstrates the effectiveness of new tailored coverage metric as an indicator of the cross-environment robustness of sensing models.

### C. Effectiveness of Input Generation

Then we evaluate the effectiveness of our input generation method by analyzing the accuracy improvement produced by our newly generately inputs.

Specifically, we use the input generation method described in Section III-C to determine a new environment to augment the test environments $E_{dev}$ and transform the sensor data in the environments. As a baseline for comparison, we include a normal input generation method which randomly picks a new environment, obtain data samples from it in the same as the existing environments in $E_{dev}$. We use both methods to generate the same amount of data and train the sensing model with the augmented datasets. The input generation effectiveness can be analyzed based on the change of accuracy and coverage produced by the generated data.

As shown in Table II, after training the sensing model with our generated inputs, the accuracy in real environments is improved by 11.78%, 15.29%, and 16.70% in all three development environment settings. As a comparison, the normal input generation method only leads to less than 4% accuracy improvement, and it may even harm the accuracy in some cases (probably due to training unstability). The accuracy in development environments $E_{dev}$ is not significantly influenced

TABLE II: The improvement of accuracy and coverage achieved with our coverage-guided input generation method. Method represents different input generation.

| $E_{dev}$ | Method | $Acc_{dev}$ | $Acc_{real}$ | Joint Coverage |
|---|---|---|---|---|
| 3 envs | Original | 97.62% | 48.45% | 15.41% |
| | Normal Input Gen | 100% | 51.80% | 23.13% |
| | Our Input Gen | 96.30% | 60.23% | 34.54% |
| 6 envs | Original | 94.05% | 49.54% | 62.26% |
| | Normal Input Gen | 96.88% | 48.45% | 62.51% |
| | Our Input Gen | 97.92% | 64.83% | 97.55% |
| 9 envs | Original | 95.24% | 59.82% | 77.83% |
| | Normal Input Gen | 97.22% | 62.82% | 77.83% |
| | Our Input Gen | 100% | 76.52% | 97.28% |

by newly generated data, which is not a problem since it is not very relevant to the real-world sensing performance.

Similarly, the environment-neuron joint coverage is improved by up to 35% after augmenting the dataset with our input generation method, while the improvement produced by normal input generation is far less significant. This demonstrates the effectiveness of our input generation method.

## V. Conclusion

We introduce to the SE community the problem of testing deep sensing systems in the smart-home industry. We highlight the unique challenge of environment dependency in such systems and propose preliminary solutions to this challenge. After demonstrating the effectiveness of our proposed environment-neuron joint coverage and coverage-guided sensing input generation, we expect there is further room for improving the test criterion design and test input generation method that can better fit the smart-home sensing system testing problem.

## Acknowledgment

This work was supported by NSFC under Grant No. 62272261 and Midea Group Co., Ltd under a collaborative research project.

## References

[1] Shuya Ding, Zhe Chen, Tianyue Zheng, and Jun Luo. Rf-net: A unified meta-learning framework for rf-enabled one-shot human activity recognition. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 517–530, 2020.

[2] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. Is neuron coverage a meaningful measure for testing deep neural networks? In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 851–862, 2020.

[3] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

[4] Xiang Li, Daqing Zhang, Qin Lv, Jie Xiong, Shengjie Li, Yue Zhang, and Hong Mei. Indotrack: Device-free indoor human tracking with commodity wi-fi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–22, 2017.

[5] Zenan Li, Xiaoxing Ma, Chang Xu, Jingwei Xu, Chun Cao, and Jian Lü. Operational calibration: Debugging confidence errors for dnns in the field. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 901–913, 2020.

[6] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.

[7] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(2):1–22, 2021.

[8] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Comput. Surv.*, 52(3), jun 2019.

[9] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[10] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[11] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. Dissector: Input validation for deep learning applications by crossing-layer dissection. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 727–738. IEEE, 2020.

[12] Wei Wang, Alex X Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. Understanding and modeling of wifi signal based human activity recognition. In *Proceedings of the 21st annual international conference on mobile computing and networking*, pages 65–76, 2015.

[13] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 775–787, 2020.

[14] Jianfei Yang, Han Zou, Hao Jiang, and Lihua Xie. Device-free occupant activity sensing using wifi-enabled iot devices for smart homes. *IEEE Internet of Things Journal*, 5(5):3991–4002, 2018.

[15] Zhou Yang, Jieke Shi, Muhammad Hilmi Asyrofi, and David Lo. Revisiting neuron coverage metrics and quality of deep neural networks. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 408–419, 2022.

[16] Daqing Zhang, Hao Wang, and Dan Wu. Toward centimeter-scale human activity sensing with wi-fi signals. *Computer*, 50(1):48–57, 2017.

[17] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142. IEEE, 2018.

[18] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 739–751. IEEE, 2020.

[19] Ziqi Zhang, Yuanchun Li, Yao Guo, Xiangqun Chen, and Yunxin Liu. Dynamic slicing for deep neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 838–850, 2020.

[20] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. Zero-effort cross-domain gesture recognition with wi-fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 313–325, 2019.

[21] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358. IEEE, 2020.

[22] Andrii Zhuravchak, Oleg Kapshii, and Evangelos Pournaras. Human activity recognition based on wi-fi csi data-a deep neural network approach. *Procedia Computer Science*, 198:59–66, 2022.