RISE: Robust Wireless Sensing Using Probabilistic and Statistical Assessments

Zhai, Shuangjiao

2021-11-25

Zhai, S, Tang, Z, Nurmi, P, Fang, D, Chen, X & Wang, Z 2021, RISE: Robust Wireless Sensing Using Probabilistic and Statistical Assessments. in MobiCom '21: Proceedings of the 27th Annual International Conference on Mobile Computing and Networking. Association for Computing Machinery (ACM), pp. 309–322, MobiCom 2021, New Orleans, United States, 28/03/2022. https://doi.org/10.1145/3447993.3483253

http://hdl.handle.net/10138/338770
10.1145/3447993.3483253

unspecified
acceptedVersion

RISE: Robust Wireless Sensing Using Probabilistic and Statistical Assessments

Shuangjiao Zhai[1], Zhanyong Tang*[1,4], Petteri Nurmi[2],
Dingyi Fang[1,4], Xiaojiang Chen[1,4], Zheng Wang*[3]
[1]Northwest University, China, [2]University of Helsinki, Finland, [3]University of Leeds, United Kingdom
[4]Shaanxi International Joint Research Centre for the Battery-Free Internet of Things
[1]sjzhai@stumail.nwu.edu.cn, [1,4]{zytang,dyf,xjchen}@nwu.edu.cn, [2]ptnurmi@cs.helsinki.fi, [3]z.wang5@leeds.ac.uk

ABSTRACT

Wireless sensing builds upon machine learning shows encouraging results. However, adopting wireless sensing as a large-scale solution remains challenging as experiences from deployments have shown the performance of a machine-learned model to suffer when there are changes in the environment, e.g., when furniture is moved or when other objects are added or removed from the environment. We present Rise, a novel solution for enhancing the robustness and performance of learning-based wireless sensing techniques against such changes during a *deployment*. Rise combines probability and statistical assessments together with anomaly detection to identify samples that are likely to be misclassified and uses feedback on these samples to update a deployed wireless sensing model. We validate Rise through extensive empirical benchmarks by considering 11 representative sensing methods covering a broad range of wireless sensing tasks. Our results show that Rise can identify 92.3% of misclassifications on average. We showcase how Rise can be combined with incremental learning to help wireless sensing models retain their performance against dynamic changes in the operating environment to reduce the maintenance cost, paving the way for learning-based wireless sensing to become capable of supporting long-term monitoring in complex everyday environments.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**.

KEYWORDS

Wireless Sensing, Machine Learning, Statistical Assessments

ACM Reference Format:
Shuangjiao Zhai[1], Zhanyong Tang*[1,4], Petteri Nurmi[2],, Dingyi Fang[1,4], Xiaojiang Chen[1,4], Zheng Wang*[3]. 2021. RISE: Robust Wireless Sensing Using Probabilistic and Statistical Assessments. In *Proceedings of MobiCom'21.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding faculty authors: Zhanyong Tang (zytang@nwu.edu.cn) and Zheng Wang (z.wang5@leeds.ac.uk).

MobiCom'21, October 2021, New Orleans

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Wireless signals like WiFi, RFID and (ultra)sound are emerging as a powerful modality for ubiquitous sensing [6, 21, 22, 27, 37, 41, 47, 50, 56, 77, 88, 94]. Indeed, wireless sensing now underpins many emerging applications, ranging from smart home personalization [16, 24] and fall monitoring [95, 99, 100] to emotion detection [98] and vital sign monitoring [14, 36, 44, 54, 63, 73].

Machine learning is an established technique for supporting wireless sensing. A machine-learned classifier is trained on a set of labeled training samples and quantifiable properties, or *features*, of the wireless signal domain. The model learns the correlation between these features and the target activity. The learned model is then applied to a test sample collected from the deployed environment. Studies show that machine-learning models can outperform approaches relied on expert-crafted analytical models [46, 76].

Despite many promising results, adopting learning-based wireless sensing as a ubiquitous solution remains challenging as experiences from deployments have shown such approaches to be sensitive to environmental changes [64, 71, 74, 81]. As we will show in this paper, even small changes in the environment can significantly degrade performance of a machine-learned sensing model. The main reason for this stems from the fact that many wireless sensing approaches rely on machine-learning techniques that inherently assume the training and test samples to come from the same or similar distribution (so-called i.i.d. assumption) [72]. Changes like moving or adding furniture, or changing the position and distance of wireless devices or the location where the activity is performed can result in a changing multi-path, bringing noise into the wireless signal characteristics. This causes the incoming test distribution to diverge from the training samples', leading to poor prediction accuracy [83]. This problem is known as *data drift* [1, 68, 101] and manifests itself as deteriorated and unreliable performance, i.e., the *robustness* of the wireless sensing solution suffers.

Existing works to improve the robustness of learning-based wireless sensing focus solely on improving model capabilities at design time. For example, data augmentation methods employ translation functions to generate synthetic or virtual training samples [15, 71, 96] or reuse knowledge from different tasks [28] to improve the generalization ability of a machine-learning model. These approaches, while important, are unlikely to cover all possible changes in the environment ahead of time [74]. Indeed, as we will show later in the paper, a data augmentation approach alone is inadequate for addressing data drift in deployments. Another solution is to find a set of features that are robust to the environment, e.g., through domain adaptation methods [83, 85]. Although this may be possible

for certain tasks and environments, it is very hard, if not impossible, to find such universal features across diverse sensing tasks and environments. The limitations of these design-time methodologies call for a different approach to address data drift.

This paper presents Rise[1], a novel approach for addressing data drift in learning-based wireless sensing and for improving sensing robustness *during deployment*. Our approach is based on *classification with rejections* [9, 39, 66], an emerging paradigm for tackling data drift by identifying and mitigating the drift as it occurs. Offering such a capability allows the deployed system to adapt to changes to improve its robustness. Here, the key and challenge of Rise are to quickly and decisively determine when data drift occurs and take action against it. Adaptation can be then achieved by either updating the model using drifting samples, rejecting drifting points where a misprediction could have a severe consequence, or choosing an alternative model capable of handling the given input. By targeting the deployment-phase, Rise thus provides an orthogonal approach for design-time solutions. It ensures the performance of an already-deployed sensing system is robust across changes and consistent over time.

Rise offers a novel and general framework for supporting learning-based sensing methods through classification with rejections. For a test input, Rise quantifies the confidence and credibility of the prediction. It then uses the quantified evaluation to recommend if we should accept the sensing outcome or if a further investigation is needed. To estimate the model's *confidence* (or certainty) in making a prediction, Rise leverages the probability distribution produced during the classification process. This approach alone, however, is insufficient as drifting samples may lead to an artificially skewed probability across class labels, resulting in falsely high confidence. To mitigate this issue, we further consider statistical evidence to assess the *credibility* (or error bound) of a prediction. We leverage conformal prediction theory [7] to quantify the non-conformity of a test input to the training samples. Our intuition is that if the test sample is significantly different from the training distribution, the predictive model will struggle to make a correct prediction because it has not acquired such knowledge during training. To detect potential mispredictions, we use the probabilistic and statistical assessment to build an unsupervised anomaly detector to approve or reject a sensing outcome based on the assessments. Our insight is that the underlying classifier is only effective when it is confident in its prediction, and its judgement is considered to be credible.

We show that Rise can effectively support two strategies for enhancing sensing robustness and helping to maintain consistently good performance over time. The first is to adopt incremental learning [2, 26] to retrain the sensing model using drifting samples collected from the deployed environment. Rise is helpful in this scenario because it only asks for user confirmation to label the drifting data but not the typical sample where the deployed model can make a correct prediction. Depending on the applications, Rise only requires user feedback to label a handful of shifting samples to adapt the sensing model to changes in the environment. The second strategy is to adopt a "mixture-of-experts" based approach [96], by employing multiple predictive models (referred to as experts) and only using predictions with high confidence and high credibility. Both strategies are orthogonal and thus can be combined.

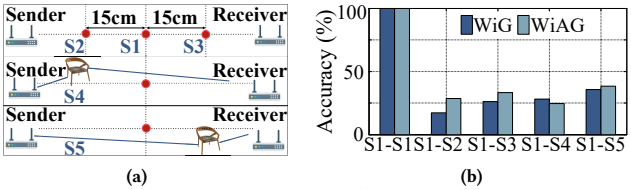


Figure 1: Wireless sensing settings (a) and impact of environmental changes (b). Dots in subgraph (a) mark where gestures were performed, and $S1 - S5$ denote different wireless setups. The x-axis of subgraph (b) denotes the training-testing setup. For example, $S1$–$S2$ means the model is trained on data collected from setting $S1$, and the trained model is tested in setting $S2$.

Results. We evaluate Rise[2] by applying it to 11 representative learning-based sensing methods [33, 40, 48, 64, 69, 71, 80, 81, 83, 84, 93]. Our large-scale case studies target various sensing tasks, ranging from gesture and identify recognition to finger input identification, covering WiFi, RFID, and sound signals as well as vibration and sensor data, and diverse machine learning algorithms (including methods based on deep neural networks [83]) and signal features. We empirically demonstrate that learning-based sensing approaches are fragile , and small changes in the environment can result in poor performance. We show that Rise is useful in detecting drifting samples, as it successfully identifies on average 92.3% (up to 100%) of the drifting samples with an average false-positive rate of 1.8%. With incremental learning, Rise can boost the sensing performance in a dynamic environment to what can be obtained in a static setting. We showcase that Rise achieves this by incurring little human involvement as it only requires user feedback to label between $1 - 3$ predicted drifting samples and using them to update the sensing model for an environmental change. The result is a new way of identifying ageing sensing models and ameliorating wireless sensing robustness and performance in the face of data drift during the deployment.

Contributions. This paper is the first to:

- exploit classification with rejections to enhance learning-based sensing robustness (Section 3);
- employ conformal prediction and anomaly detection to detect data drift (Section 3);
- combine incremental and ensemble learning and anomaly detection to improve sensing performance (Section 3.4).

2 BACKGROUND AND MOTIVATION

2.1 Problem Scope

Our work focuses on improving the robustness of wireless sensing techniques that rely on a machine-learning classifier during *deployment*. Many of such solutions rely on wireless signal characteristics that are inherently sensitive to environmental changes. We note that there are sensing systems build on techniques like the frequency-modulated continuous-wave (FMCW) [3, 52, 79, 92] or physiological data [17, 89]. Those approaches are robust to

[1] Rise =**R**obust w**I**reless **SE**nsing.

[2] Code and data are available at: https://github.com/NWU-NISL-Sensing/RISE.

changes in the spatial domain. Our work does not target those environmental-agnostic sensing techniques.

2.2 Impact of Environmental Changes

To illustrate the impact of environmental changes, we consider WiFi-based gesture recognition as a representative task. Later in the paper, we extend our experiments to other types of wireless sensing tasks. We consider two learning-based gesture recognition methods, WiG [33] and WiAG [71] which use different predictive modeling techniques and wireless channel characteristics. The latter method is also a data augmentation approach that uses virtual samples to improve the generalization ability of the sensing mode. We apply each method to six representative gestures, including "push and pull", "draw a circle", "throw", "slide", "sweep" and "draw zigzag". The gestures were performed in a controlled environment within a radio frequency anechoic chamber to minimize the impact of other parameters. We consider the five wireless settings depicted in Figure 1a. Here, a dot represents a location where the gestures were performed (denoted as S1, S2, S3). We also placed a chair at two locations S4 and S5 to mimic multipath due to environmental changes. To ensure the gestures were done consistently, we use a programmable robotic arm to perform the target gestures. More details of the experimental setup are given in Sections 4 and 5.4.1. Even though the experiments were conducted in a highly restricted environment, a change in the environment can lead to frequent misclassification of a sensing model.

Figure 1b shows what happens if we first train the model using data collected from S1 and apply the learned model to samples obtained in each setting. We also used the location-angle translation function of WiAG to generate virtual samples for S2 and S3. Since we use a robotic arm to perform gestures, there is little variance between activities of the same gesture. Therefore, both models achieve 100% accuracy when the test data is collected from S1. However, the accuracy of WiG and WiAG drops to less than 40% when they are tested on samples collected from other settings, showing that the environmental changes can have a severe impact on the sensing performance.

As we will show later, while using training data collected from multiple environments can improve the model reliability, the performance of a deployed model can still suffer from small changes that were hard to anticipate ahead during the design phrase.

2.3 Evaluating Model Credibility

To identify unreliable sensing outcomes, a naïve solution would be to consider the probability given by a classifier. For example, a high prediction probability of a chosen label can suggest that the model is more certain about its prediction. While the probability indicates the model's confidence in making a prediction, it is insufficient for measuring the model's credibility due to the way supervised learning works. Machine learning classifiers compute the probability of a sample fitting into a class for predictions, but a drifting sample can lead to a skewed probability distribution.

As an intuitive example, consider a binary classification problem where the sensing model needs to predict if the test input belongs to one of the two activities (e.g., gestures or human subjects) or classes - c_1 and c_2. When making a prediction, the model estimates the probability of the input belonging to each class. Let r^1 and r^2 denote the probability that the input belongs to c_1 and c_2 respectively. If the input pattern is not seen during the training phase, the model could give a low probability score, r^1 for c_1, such that, $r^1 \sim 0.0$. This would artificially push the probability of another class, r^2, towards 1.0 as the sum of the probabilities must be 1.0 [10]. To estimate the model's credibility, we need a way to assess if the input is likely to be compatible with the knowledge the model learned from its training data. For this specific example, we wish to assign a low credibility score to both classes if the input sample is highly distinct from those seen at the training stage.

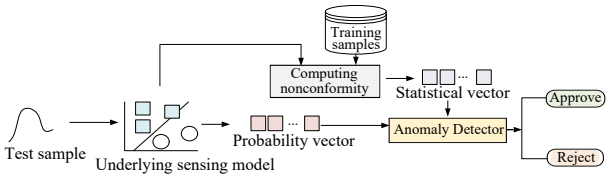


Figure 2: We feed the probability vector given by the underlying model and the statistical vector given by a nonconformity function to an anomaly detector to approve or reject a sensing outcome.

2.4 Terminologies

In this work, we measure the model *confidence* by implicitly computing the significance of the probability given by a machine-learning classifier. This metric is computed from a probability vector described in Section 3.1. We note that our definition of confidence is different from the *statistical confidence interval* that estimates the range where the true value of a parameter may lie in. Furthermore, we use a statistical vector, described in Section 3.2, to measure the *credibility* of a model prediction. Our credibility metric is different from the classical credibility definition in the CP theory where a single scalar value is used as the credibility measurement. Unless state otherwise, we use our definitions in the rest of the paper.

3 OUR APPROACH

Figure 2 gives a high-level view of Rise that works for any probability-based machine-learning classifier – a dominating approach for wireless sensing [11, 12, 23, 40, 43, 45, 49, 62, 64, 71, 78, 82, 83, 90, 91, 97, 100]. At the core of Rise is a mechanism for quantifying both the *confidence* (Section 3.1) and *credibility* (Section 3.2) of a classification outcome. This evaluation is fed into an anomaly detector (Section 3.3) to determine if we should accept a sensing outcome. We stress that Rise does not require changing the underlying sensing model - the input and output of a sensing model will be in the same format as when the model is used alone. Instead, Rise only takes as input the intermediate results (e.g., the probabilities produced by the sensing model).

3.1 Probabilistic Assessment

To evaluate how confident a classification-based sensing model predicts a given input, we examine the probability distribution of the prediction. Our insight is that the more significant the probability the chosen class is with respect to others, the higher confidence the classifier will be (ignoring the credibility for now). If several labels have more or less the same prediction probabilities, the model would struggle in choosing from these candidate classes.

To elaborate on our observation, consider a learning-based sensing system that uses a classifier to attribute the wireless signal to one of the five targeting activities (or prediction classes), c_1, c_2,

Table 1: Nonconformity measures supported by Rise

ML algorithm	NCM	Description
k Near Neighbor (KNN)	$\frac{\sum_{j=1}^{k} D_{sj}^{y}}{\sum_{j=1}^{k} D_{sj}^{-y}}$	The ratio of the sum of the distances of k nearest neighbors in the same class as the predicted label y of sample s to the sum of the distances of k nearest neighbors in all other classes [72].
Support Vector Machine (SVM)	Lagrange Multipliers	Extract Lagrange multipliers by maximizing $\sum_{i=1}^{l+1} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l+1} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$, where, K refers to the kernel function. Lagrange multipliers can reflect the "strangeness" of a sample, and so can be interpreted as the NCM [7].
Random Forest (RF)	$-\frac{\lvert i \in \{1,\cdots,n\}, T_i(x_s)=y \rvert}{n}$	The percentage of predicted label y for sample s given by decision trees. T_i is the $i-th$ decision tree in the RF [20].
Logistics Regression (LR)	$(-1)^{y}(\theta - x_s \cdot \omega)$	This applies mainly to binary classification, if $x_s \cdot \omega < \theta$, $y = 1$, otherwise $y = 0$. Here, θ refers to a parameter and ω refers to a weighting vector of attributes [8].
Any probabilistic model	$1 - \left\{ \frac{prob_s^{y} - max\left(prob_s^{-y}\right)}{2} \right\}$	Here, $prob_s^{y}$ is the probability of sample s being classified as class y, and $prob_s^{-y}$ is the probability of sample s being classified as all other classes [61]. This is a generic NCM that works for any probabilistic ML model.

c_3, c_4, and c_5. Suppose the probability distribution for samples a and b is {**0.51**, **0.48**, 0.003, 0.003, 0.004} and {**0.51**, 0.12, 0.12, 0.12, 0.13}, respectively. Both probability distributions sum to 1.0. In this example, the two test samples will be attributed to class c_1, because this class label has the highest probability of 0.51 in the relevant prediction. Existing learning-based method would simply use c_1 as the sensing outcome. If we look closely at the probability distribution, we can see that the model is less certain in predicting sample a than b, because the probability difference between c_1 and c_2 is less significant in the first prediction (0.51 vs 0.48) than the second one (0.51 vs 0.12).

In light of this observation, we use the probability distribution to estimate the model's confidence in making a prediction. To this end, we feed the probability vector given by the underlying classifier to an anomaly detector to correlate the probability distribution with the prediction confidence.

3.2 Statistical Assessment

To evaluate the model's credibility, we leverage the conformal prediction (CP) theory [61]. This method computes the level of credibility for a prediction by using past experience to account for data drift. CP works like the statistical confidence interval. Given an underlying classifier, g, and a significance level, ε, CP produces a prediction region: a candidate label set that is guaranteed to contain the correct label with probability no more than $1 - \varepsilon$. Essentially, ε defines the sensitivity of prediction region. The larger ε is, the smaller prediction region will be. To generate the label set, CP uses a *nonconformity measure* derived from g to quantify how *dissimilar* the input sample is compared to a history of past examples (e.g., training examples). Intuitively, the more the model input deviates from the training examples, the lower credibility the model's prediction is likely to be. Thus, a low credibility suggests that there is a high chance that the corresponding example is drifting with respect to training examples. Such an example is at risk of being misclassified due to limited knowledge of the underlying model.

Nonconformity measures. We use a nonconformity function, f, to score how different a sample s is from a set of previous examples, $B = \{s_1, s_2, \ldots, s_n\}$, $a_s = f(B, s)$. The greater the value of the nonconformity measure (NCM), a_s, the less similar s is to the elements of B. In this work, B is referred to as the *calibration set* and it contains the samples seen by the underlying sensing model during the training phase. Rise provides a range of *ready-to-use* functions to compute the NCM, listed in Table 1. Some of the functions are specific to the machine-learning technique used, by utilizing the internal algorithmic information. By default, we use a model-specific nonconformity function if it is available. However, we also provide a general nonconformity function to be used by any other probabilistic-based classifier. Note that the nonconformity function only depends on the underlying classifier and not on the sensing task, the model features or the signal representation that is used.

Compute the p-value. We construct a conformal evaluator to approve or reject a null hypothesis asserting that the input sample s does not belong in the prediction region formed by elements of B. To test the hypothesis, we compute a *p-value* using the chosen nonconformity function to measure how similar the new sample to all data points in the calibration set. We first use the nonconformity function f to compute, *offline*, the nonconformity score, $a_1^{y}, a_2^{y}, \ldots, a_n^{y}$, for each of the n instances in the calibration set, B, to each of the supported class labels, y (e.g., a target activity type) using the underlying sensing model. After deploying the sensing model, we calculate the nonconformity score, $a_{n+1}^{y^p}$, for a new test sample, s, belongs to a target label y^p. We then compute the *p-value*, p_s, for test sample s as:

$$P_s = \frac{\text{COUNT}\left\{i \in \{1, \ldots, n+1\} : y_i = y^p \text{ and } \alpha_i^{y^p} \geq \alpha_{n+1}^{y^p}\right\}}{\text{COUNT}\ \{i \in \{1, \ldots, n+1\} : y_i = y^p\}} \quad (1)$$

Essentially, we count the proportion of calibration samples for label y^p with a nonconformity score greater or equal to the nonconformity score of sample s belongs to class y^p. If the p-value is small (close to its lower bound $1/(n+1)$), the prediction is very *nonconforming*, meaning the nonconformity score for most of the examples in the n-element calibration set is smaller than a_{n+1}. In other words, the test sample is very dissimilar to the past samples seen during training. If the p-value is large (close to its upper bound 1), then the prediction is very conforming, suggesting the input sample is similar to most of the training samples of label y^p.

Applications of CP in other domains [39, 55] use a single p-value for decision making - if p_s falls above a given significance level, ε, the model's prediction y is accepted as a valid prediction. We take a different approach. Instead of setting a fixed threshold for ε, we use a statistical, unsupervised method to build an anomaly detector to approve or reject the sensing model's prediction (Section 3.3).

Calibration datasets. To set aside a calibration dataset to compute the p-value, we partition the training set into k folds of equal size. We leave one fold as the calibration set for computing the nonconformity score on the calibration samples (by training an underlying sensing classifier on the remaining $k-1$ folds). We repeat this process k times until all folds have been used as the calibration set once. This results in k underlying classifiers (used by CP only). During deployment, we use the k underlying classifiers and their corresponding calibration sets to compute k p-values. We then take the average p-value as the final output. In this work, we empirically set k to 10. Note that this process is automated and the

developer of the wireless sensing solution simply needs to call the API of Rise; see Section 3.5.

The statistical vector. For a given sample, s, we compute the p-value when attributing it to each of the supported class labels. The resulting p-values then form a statistical vector, where each element of the vector is a p-value for a class label. The statistical vector is used to measure the "strangeness" of the sample to those seen at the training stage for each supported class. Like the probability vector, using a statistical vector rather than a single p-value allows us to capture the certainty of CP in evaluating the credibility.

3.3 Detect Drifting Samples

We concatenate the probability and the statistical vectors as one input to an anomaly detector to approve or reject a sensing outcome. Our anomaly detector is a one-class (or one-vs-all) support vector machine (SVM) constructed from the 'normal' data - the sensing model's training samples in our case. It learns the boundaries of the training samples and can classify points that lie outside the boundary, i.e., the outliers. We choose this algorithm as it is shown to be effective in identifying a skewed data distribution [60, 67].

The key idea of abnormal detection is to detect rarely seen events. This is achieved by finding a closeness boundary in a feature space such that a data point outsides the boundary is considered an outlier. In our case, the feature space is defined by the probabilistic and statistical vectors described in Sections 3.1 and 3.2. In the context of the one-class SVM, the boundary is a hyperplane for separating the normal (i.e., sensing model training samples) data points from the origin in the feature space - such that the hyperplane border is as close to the normal data points as possible. During deployment, our abnormal detector checks if the input test sample is within the boundary constructed from sensing model training samples. If not, the test sample is considered an outlier (i.e., a drifting sample). For more information about abnormal detection, we refer to [34].

For a sensing model that supports n targets, $c_1, c_2, \cdots, c_n$, its underlying classifier will give a probability vector of n elements of probabilistic values $\{r^1, r^2, \cdots, r^n\}$. Similarly, our conformal evaluator will produce a statistical vector of n elements for the n target activities $\{p^1, p^2, \cdots, p^n\}$. This will result in a $2n$ input vector, $\{r^1, r^2, \cdots, r^n, p^1, p^2, \cdots, p^n\}$, given to the anomaly detector which outputs a binary value for approving or rejecting a sensing model prediction. Rise *automatically* constructs a one-class SVM for each of the supported class labels. Specifically, the probability vector and statistical vector of each training sample are calculated using k-fold cross-validation, and then the one-class SVM is trained by using the probability vector and statistical vector of the training sample of the same label. For a sensing system that supports n activities, we thus will have n one-class SVM models $M^{c_1}, M^{c_2}, \cdots, M^{c_n}$.

During deployment, for a test sample s_t, we use the sensing model's prediction c_{s_t} to choose the appropriate SVM model $M^{c_{s_t}}$, which takes in the joint probability-statistical vector, $\{r^1_{s_t}, r^2_{s_t}, \cdots, r^n_{s_t}, p^1_{s_t}, p^2_{s_t}, \cdots, p^n_{s_t}\}$, to approve or reject the sensing outcome. Rejected sensing outcomes can be quarantined and dealt with separately, either warranting manual inspection or checking through other means. Although handling drifting samples raises several other challenges (e.g., how to correct the sensing outcomes through user feedback), we note that such remediation is only possible once drifting samples are detected - the main focus of this paper.

Table 2: Sensing methods evaluated in this work.

Sensing Method	Features	ML Model
WiG [33]	Statistical measures	SVM
WiAG [71]	Discrete Wavelet Transform (DWT)	K-nearest neighbor
TACT [80]	Dynamic Time Warping (DTW) measures	Random forests (RF)
AllSee [40]	Amplitude	K-nearest neighbor
EI [83]	Neural network representations	Convolutional Neural network (CNN)
WiWho [93]	Statistical measures	Decision Tree (DT)
WifiU [81]	Spectrogram	SVM
VibWrite [48]	Mel Frequency Cepstral Coefficents (MFCC)	SVM
Taprint [84]	Fast Fourier Transform (FFT)	K-nearest neighbor
UDO-Free [69]	Statistical measures	K-nearest neighbor
M-Touch [64]	Physiological characteristics	K-nearest neighbor

3.4 Improve Sensing Robustness

Rise can be used during deployment with incremental learning to improve a sensing model's performance or with ensembling learning to enhance a model's robustness.

Incremental learning. Rise can improve the performance of a sensing system in the deployed environment. This can be achieved through incremental learning [2, 26], where we isolate the data samples filtered out by Rise for further inspection. By manually labeling the drifting samples and adding them to the existing training dataset, we can then retrain the underlying model and update the Rise anomaly detector on the new samples. Note that we also update the calibration dataset and the anomaly detector when new training samples are added. This allows the Rise decision process to adapt to the change of the underlying model. Such an approach minimizes user interruption by only asking for user feedback when the data drift is likely to happen. This feature is vital for minimizing the impact of user experience and reducing the associated maintenance cost. Later in Section 5, we show that Rise can effectively support incremental learning.

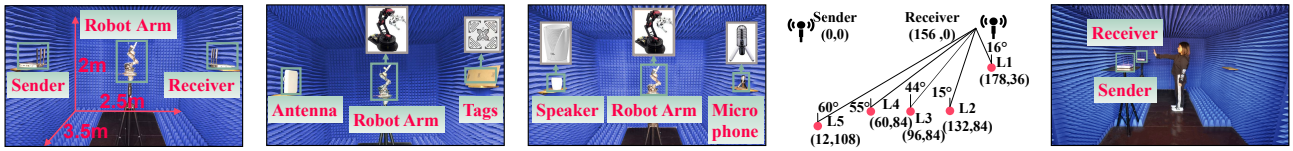
Ensemble learning. Another option is to train a set of classifiers and aggregate the credible predictions given by different classifiers as the outcome. Such a strategy is also known as mixture-of-experts (or stacking based ensemble learning) [53, 96], where several heterogeneous models are trained to collectively solve the same problem. The key challenge here is that different algorithms have different capabilities in learning and modeling, and some may only be effective in certain types of samples. To minimize the impact of poor advice given by an ineffective model, we can use the Rise anomaly detector to filter out low-quality predictions. To this end, our SVM produces a distance metric, measuring how likely each prediction deviates from the normal distribution. We can then choose the $top-n$ predictions with the smallest distance and apply majority voting to produce a final classification. In Section 5.3.1, we show that Rise can effectively improve performance of ensemble learning.

3.5 Implementation

We developed an open-source prototype of Rise using Python and the scikit-learn package [65]. To use Rise, the developer needs to choose a nonconformity measure and use the Rise APIs to *automatically* construct the calibration dataset and the anomaly detector from the sensing model's training samples (see Section 3.2). We note that using Rise does not require making any changes to the wireless sensing model and Rise can simply be used as an additional support tool. During deployment, for a given test sample, the sensing system will pass the model's underlying algorithm and its prediction as arguments to a dedicated Rise API. The API will

Table 3: Case studies and their setups

Case Studies	Sensing Tasks	Signal	Sensing Method	Evaluation Site	Participants	Config. Labels	Activity (#types)	#Samples	Accuracy (%)
Case study 1	Gesture recognition	WiFi	WiG	Controlled environment	Robotic Arm	S1, S2, S3, S4, S5	Gestures (6)	900	100↑8
			WiAG						100↑8.6
			WiAG-C	Controlled environment	6 volunteers	L1, L2, L3, L4, L5	Gestures (6)	5,400	99.9↑8.5
			WiAG-O	Office	6 volunteers	L1, L2, L3, L4, L5	Gestures (6)	5,400	93.9↑2.5
		RFID	TACT	Controlled environment	Robotic Arm	S1, S2, S3, S4, S5	Gestures (6)	900	98.4↑4.9
			AllSee						98.8↑1.8
		Ultrasound	EI	Controlled environment	Robotic Arm	S1, S2, S3, S4, S5	Gestures (6)	900	99.8↑20
Case study 2	Gait recognition	WiFi	WiWho	Controlled environment	15 volunteers	S1, S2, S3, S4, S5	Gaits (15)	2,250	95↑3
			WifiU						97.6↑4.55
Case study 3	Activity recognition	Vibration	VibWrite-R	Office	15 volunteers	N, 1, 2, 3, 4	Finger Inputs (10)	15,000	99.5↓0.5
			VibWrite-A		15 volunteers	N, 1, 2, 3, 4	User identification (15)	15,000	98.7↑3.7
			Taprint-R	Outdoor	15 volunteers	M	Text Inputs (12)	9,000	99.4↑3.4
			Taprint-A	Office	15 volunteers	S, D, W	User identification (15)	27,000	99.1↑1.5
		IMU Sensors	UDO-Free	Outdoor	15 volunteers	RP, LP, DC, DB	Activities (3)	18,000	98.8↑0.68
				Office	15 volunteers	RP, LP, DC, DB	Activities (2)	12,000	
			M-Touch	Outdoor	15 volunteers	M	User identification (15)	450	97.9↑1.08
				Office	15 volunteers	S, DP, T	User identification (15)	1,350	



(a) WiFi (robotic arm) (b) RFID (robotic arm) (c) Ultrasound (robotic arm) (d) Five locations of WiAG (e) WiFi (human gestures)

Figure 3: Controlled environment setup for gesture recognition using various signals.

(a) WiFi sensing (office) (b) Vibwrite (office) (c) Taprint (outdoor & office) (d) UDO-Free (outdoor & office) (e) M-Touch (outdoor & office)

Figure 4: Other evaluation setups when replicating the source publications.

then compute the probability and statistical vector to feed into the anomaly detector to approve or reject the sensing outcome.

4 EXPERIMENTAL SETUP

4.1 Case Studies

We perform a large-scale study by applying RISE to 11 representative learning-based sensing methods [33, 40, 48, 64, 69, 71, 80, 81, 83, 84, 93]. As summarized in Table 2, we organize the tests around three case studies. These methods include contact and non-contact sensing applications, covering a wide range of wireless signals, sensing tasks, machine learning algorithms and features. Our test methods also include a representative data augmentation approach [71] that uses virtual samples to improve sensing performance, as well as methods that use environment-agnostic features like [81].

We reproduced the test methods by faithfully following the methodology described in their source publications, and use the relevant open-source code implementation when available. Table 3 gives the accuracy obtained by our reproduction in a static environment, with a comparison to the results reported in the source publication. Here, the performance improvement or degradation is marked by a ↑ or ↓ symbol respectively. As can be seen from the table, our reproduction gives comparable (and often better) results compared to the numbers reported in source publications.

We also note that the quality of the underlying machine-learning model of a sensing system can greatly impact the performance of RISE. However, we expect the developers make efforts to build a reasonably good model from the initial training dataset.

4.2 Testing Environments

We collect the raw wireless signals in both a controlled environment and day-to-day scenarios. As shown in Figure 3, our controlled environment is a radio frequency (RF) anechoic chamber of $3.5m \times 2.5m \times 2m$. The chamber is designed to reduce multi-path within the room because the vast majority of the reflected signal will be absorbed when the signal bounces to the wall, ceiling, and the ground [5, 32]. We use the RF chamber to control the environmental parameters to ensure the reproducibility of experimental runs by cancelling uncontrolled multi-paths that can have an impact on activity recognition [81]. In addition to this, we also conducted experiments in an office and outdoor settings as depicted in Figure 4.

4.3 Sensing Tasks and Notations

Table 3 summarizes how we test for different sensing tasks.

Gesture recognition. For this case study, we consider six representative gestures, including "push and pull", "draw a circle", "throw", "slide", "sweep" and "draw zigzag" that are commonly supported by prior methods [33, 57, 71, 91]. From the controlled environment (Figure 3), we collected WiFi, RFID and Ultrasound signals. For reproducibility, we use a programmable robotic arm to perform the gestures. We have five settings in the controlled environment, marked as S1 to S5 in Figure 1a. For WiAG (a data augmentation

Table 4: Summary of changes introduced in our evaluation

Sensing Tasks	Sensing Method	Config. Labels	Description
Gesture recognition	WiG WiAG	S1, S2, S3, S4, S5	We changed the location where the gestures were performed ($S1$, $S2$, $S3$) and added chairs in different positions to mimic multi-path ($S4$, $S5$) when the gestures were performed at location $S1$. See also Figure 1a and Section 5.4.1.
	WiAG-C WiAG-O	L1, L2, L3, L4, L5	Gestures were performed in five different locations ($L1 - L5$) per the evaluation settings used in the source publication of WiAG [71]. See also Figure 3d and Section 5.4.1.
	TACT AllSee EI	S1, S2, S3, S4, S5	We changed the location where the gestures were performed ($S1$, $S2$, $S3$) and added chairs in different positions to mimic multi-path ($S4$, $S5$) when the gestures were performed at location $S1$. See also Figure 1a and Section 5.4.1.
Gait recognition	WiWho WifiU	S1, S2, S3, S4, S5	We asked participants to walk along different paths ($S1$, $S2$, $S3$) and added chairs in different positions to mimic multi-path ($S4$, $S5$) when participants were walking along path $S1$. See also Figure 1a and Section 5.4.2.
Activity recognition	VibWrite	N, 1, 2, 3, 4	We ensured that the sensing area was free of obstacles (N) or placing a glass of water at four different locations (1, 2, 3, 4). See also Figure 4b and Section 5.4.3.
	Taprint	S, D, M, W	This case study involves multiple indoor and outdoor settings with dry and wet hands. For the indoor setting, we asked participants to tap the 12 knuckles (Figure 4c) on their hands with standard force (S) and a different force (D) when the hand is dry or wet (W). Then, we asked participants to tap the 12 knuckles with standard force during the outdoor setting when walking with a dry hand (M). See also Section 5.4.3.
	UDO-Free	RP, LP, DC, DB	We asked participants to put the mobile phone in their right and left pockets (RP, LP) to perform the set of actions supported by UDO-Free [69]. Then, we asked participants to place the phone in the right pocket to perform the "Sitting" and "Biking" actions using different chairs (DC) and bicycles (DB). See also Figure 4d and Section 5.4.3.
	M-Touch	S, DP, T, M	We first asked participants to hold the smartphone in their hands and perform an action in different postures, including sitting (S), standing (DP), and walking (M). Then, we asked our participants to perform the action in a sitting posture while the phone was placed on the table (T) as illustrated in Figure 4e. See also Section 5.4.3.

method), we collected WiFi signals from the controlled environment (Figure 3e) and the office (Figure 4a). In addition to evaluating WiAG under the S1 to S5 settings in the controlled environment, we also create two evaluation variants, WiAG-C and WiAG-O. We follow the experimental setup of WiAG's source publication, using gestures data from five settings (L1 to L5 in Figure 3d), where we give the coordination and angle to the sender (0,0). We denote the evaluation from the controlled environment and the office as WiAG-C and WiAG-O for L1 to L5, respectively.

Gait recognition. For this task, we collected gait data from 15 volunteers (8 females) under the S1 to S5 settings from the controlled environment using WiFi signals.

Activity recognition. For this task, we replicated the setup of the tested methods (Figure 4 b-e). We use VibWrite-R and Taprint-R to denote input recognition of VibWrite and Taprint, respectively and VibWrite-A and Taprint-A to denote user identification of VibWrite and Taprint, respectively. The user identification method of VibWrite and Taprint predicts which of the target users entered the texts. UDO-Free detects five human activities of "Running", "Walking", "Biking", "Standing" and "Sitting", using data collected from inertial measurement unit (IMU) sensors of a smartphone. M-Touch also targets user identification using data collected from the IMU sensors. Our participants are the same 15 users for gait recognition.

4.4 Device Setup

WiFi. We use two mini PCs, each has an Intel 5300 network interface card (NIC), as the sender and receiver to collect WiFi signals (Figure 3a). We use an open-source channel state information (CSI) measurement tool [31] to collect the CSI measurement. Following the common practice of prior work [91, 96], we configure the sender to send 1,000 packets per second to the receiver.

RFID. We use the H47 RFID tag and a directional antenna powered by an Impinj R420 RFID reader to transmit and receive signals (Figure 3b). We use the same number of tags as the source publications of the relevant methods, 4 for TACT [80] and 3 for AllSee [40].

Ultrasound. We use a commercial speaker (JBL Jembe) to transmit a modulated 19 kHz ultrasound and a microphone (SAMSON MeteorMic) to collect the ultrasound signals (Figure 3c). The speaker and microphone are connected to a laptop for data processing.

Vibration. We replicate the setup of Vibwrite [48] by connecting a vibration motor and a piezoelectric sensor to a laptop to send and receive vibration signals (Figure 4b).

Sensor data. For finger inputs (Figure 4c), we asked our 15 volunteers to wear a LR G smartwatch on their left hand and tap each of the 12 knuckles (on the left hand) 50 times with their right hand. We use the IMU of the smartwatch to collect sensor data. We also use a Xiaomi smartphone that runs a customized Android app to collect IMU sensor data for other tasks (Figures 4d and 4e).

4.5 Environmental Changes

As summarized in Table 4, we added various changes to the training-testing setups by either changing the locations and ways an activity is performed or adding furniture to introduce additional multi-path. To visualize the impact of environmental changes to signal data, we apply t-SNE [70] to project the data samples collected from each setup to a two-dimensional space for each tested method. As can be seen from Figure 5, data points collected from the same environment sit nearby in the projected area, and samples from different settings tend to group as separate clusters. This suggests that the environmental changes introduced indeed impact the data distribution of the wireless samples.

4.6 Evaluation Methodology

Our evaluation is designed to answer two questions. The first is whether Rise can effectively filter out drifting samples (Section 5.2). The second is if Rise can be used to improve the robustness of the deployed sensing model (Section 5.3).

4.6.1 Model evaluation. We consider two training-testing scenarios: *static* and *dynamic*. In a static environment, the training and testing samples come from the same setup. In a dynamic setting, we mix data samples from different settings, including the ones used to collect the sensing model training data and those from other environmental settings. To mix the data from the training

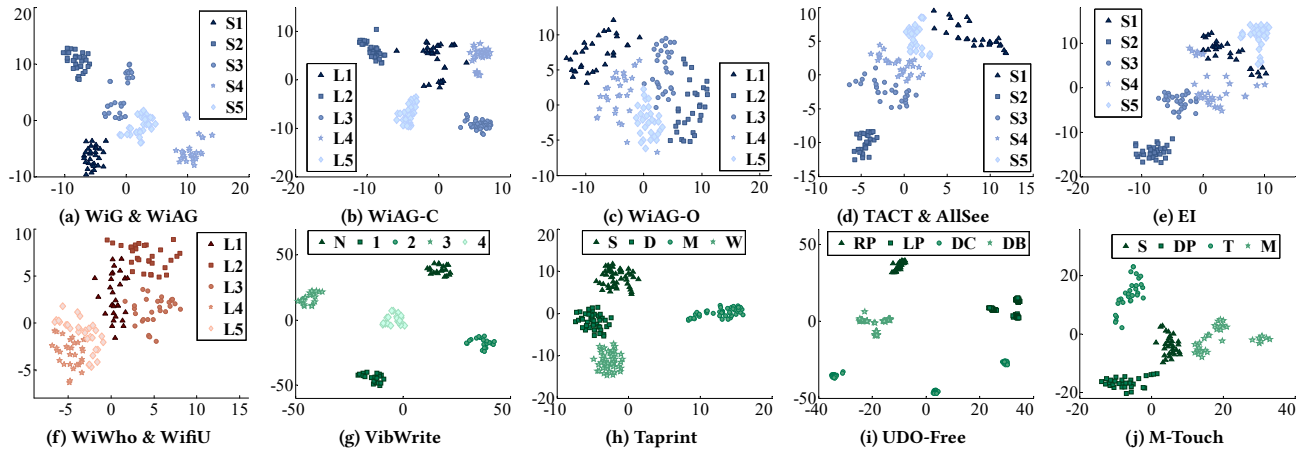


Figure 5: Visualization of data samples collected from different evaluation settings per method using t-SNE [70]. Data samples collected from different setups tend to lie in separate data clusters, suggesting that the environmental changes impact the wireless sample distribution.

environment, we apply 5-fold cross-validation to the data obtained from the training environment. This means we partition the data into 5 sets; we train the sensing model on four sets and then test the trained model on the remaining set (together with data from the dynamic setting). This is a standard evaluation methodology, providing an estimate of the generalization ability of a machine-learning model in predicting unseen data. We introduce the changes in the dynamic setting by simulating the commonly seen changes in a real-life deployment, such as adding or removing objects and changing the position where an activity is performed. Note that we apply cross-validation [13] to ensure each setup is used both as training and testing. We then report the average performance and variants across cross-validation setups.

4.6.2 Metrics. We consider the following "*higher-is-better*" metrics for detecting drifting samples:

Accuracy. The ratio of the number of correctly predicted samples to the total number of testing samples.

Precision (P). The ratio of correctly predicted samples to the total number of samples that are predicted to have a label. This metric answers questions like "*Of all detected drifting samples, how many are correct?*". High precision indicates a low *false-positive* rate, meaning Rise rarely mis-classifies normal samples as drifting.

Recall (R). The ratio of correctly predicted samples to the total number of test samples that belong to a class. This metric answers questions like "*Of all drifting samples, how many are actually detected by Rise?*". High recall suggests a low *false-negative* rate, indicating Rise can identify most of the samples that a sensing model will mis-predict.

F1-score. The harmonic mean of Precision and Recall, calculated as $2 \times \frac{Precision \times Recall}{Precision + Recall}$. A high F1-score means Rise can detect most drifting samples while rarely mis-classifying normal samples.

5 EXPERIMENTAL RESULTS

Our evaluation shows that all tested approaches (including data augmentation [71] and feature engineering [81] methods) suffer from dynamic changes during deployment (Section 5.1). Rise can identify 92.3% of the drifting samples (Section 5.2). When using

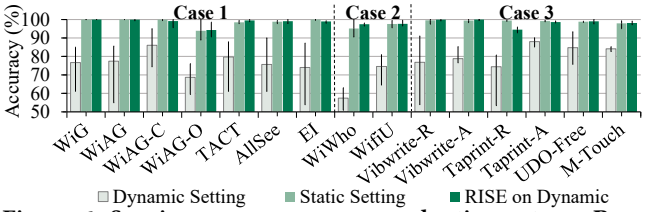


Figure 6: Sensing accuracy across evaluation setups. Rise can help filter out drifting samples and retain the sensing performance with incremental learning.

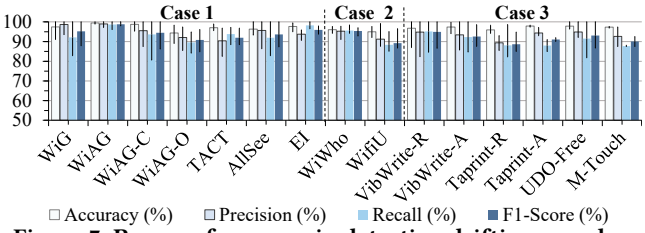


Figure 7: Rise performance in detecting drifting samples.

with incremental learning, Rise improves the performance of a deployed model by 21% on average, using *at most* three identified drifting samples to update a deployed sensing model (Section 5.3.1).

5.1 Overall Results

Figure 6 shows the sensing accuracy in the static and dynamic environment. Here, the min-max bar shows the variances across cross-validations (Section 4.6.1). The implementation variants (WiAG-C/O, Vibwrite-R/A and Taprint-R/A) were defined in Section 4.3.

In a static environment, all sensing methods achieve a prediction accuracy of over 93%. However, their performance can suffer in a dynamic setting[3], where we observe a average decrease of 22% in the prediction accuracy. The impact of drifting samples is also noticeable for wireless signals like WiFi and Ultrasound, where we see a drop in the prediction accuracy of over 40%. This is not surprising as these signals are known to be sensitive to changes

[3]Since we apply cross-validation to include data collected from different environment settings in a dynamic evaluation setup, we provide a diverse set of training data to train the underlying sensing model to avoid over-fitting.

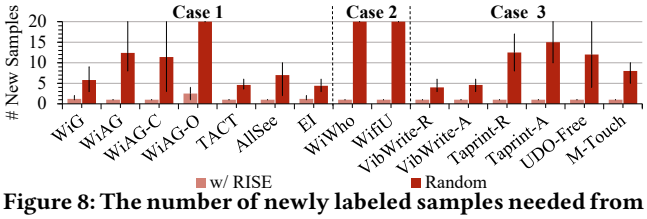


Figure 8: The number of newly labeled samples needed from a dynamic environment to retain 95% of the static performance of the underlying model.

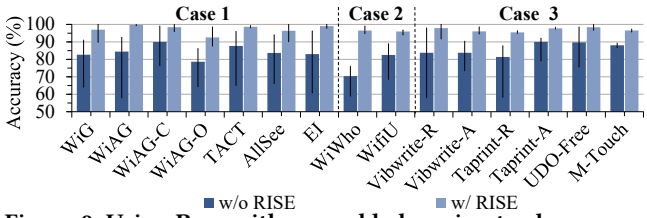


Figure 9: Using Rise with ensemble learning to choose predictions from a bag of machine learning models.

[51, 71, 83, 91]. By applying Rise to filter out drifting samples and use a handful of labeled drifting samples to update the sensing model, one can retain the accuracy of the relevant method to a level close to its performance in a static environment. We also see that the performance of Rise is stable, regardless of the underlying machine learning or wireless signals.

5.2 Detecting Drifting Samples

Figure 7 shows the performance of Rise for predicting drifting samples. For most (80%) of the sensing methods, Rise gives an average precision of 92% (at least 89%), with an average accuracy of 96% (at least 94%) for a given method. This means it rarely filters out correct sensing predictions. For a few sensing methods, like Taprint-R, Rise gives an average precision of 89%. This translates to a false-positive rate (i.e., mis-labelling a normal smaple to be drifting) of 3.7%[4], meaning Rise can sometimes reject correct sensing predictions. We found that this is limited by the underlying model's capability in a dynamic environment. When the underlying model performs poorly in a dynamic environment, its prediction probability vector becomes noisy, which in turns affects Rise in filtering out drifting samples. We also observe a similar trend for Recall and the F1-score, where Rise is effective for most of the sensing methods with a Recall or F1-score greater than 90%, except for WiAG-O, WifiU, Taprint and M-Touch, where Rise can miss some drifting samples. Overall, Rise gives a precision of 94.5% and a Recall of 92.3%, averaged across case studies. The results translate to a false-positive rate of 1.8% and a false-negative rate (i.e., missing a mis-prediction) of 7.7%.

We note that the severity of the drift is a subjective matter. For critical applications, even a few misclassifications can cause major issues. For this reason, Rise allows the developer to configure the significant level, ε, of our CP model to control the number of samples to be manually inspected. As confidence is computed as $1-\varepsilon$, a smaller ε will lead to more frequent false-positive, but this can reduce the impact of a misprediction. In this work, we empirically set ε to 0.1 by applying cross-validation to the calibration dataset.

5.3 Improving Sensing Methods

We have shown that Rise is effective in detecting drifting samples. We now demonstrate two ways of using Rise to improve sensing performance and robustness.

5.3.1 Incremental learning. Figure 8 shows how many new samples are needed from the deployment environment to update the model to achieve 95% of its performance in a static environment. For fairness, we make sure that all approaches are tested on the same number and set of *unseen* drifting samples. Without Rise, one would have to first randomly choose test samples to label and then add the labeled samples to the existing training dataset to update a deployed model. Such a random strategy has poor user experience or high maintenance cost by asking the user or the system developers to confirm or label activities where the underlying model can successfully process. Rise avoids this pitfall by only asking for user intervention when it thinks this is necessary. To improve the underlying model's performance, Rise on average only requires labeling 1.12 (often just one) of the samples that are predicted to be drifting. This represents a reduction of 89% of test samples required user intervention. We remark that a critical benefit of Rise is it can automatically detect when a model update is needed. This capability reduces the human effort, by only seeking user involvement only when data drifting incurs.

5.3.2 Ensemble learning. In this experiment, we apply Rise to the ensemble learning strategy described in Section 3.4. Our ensemble classifier committee includes 11 representative linear and non-linear classifiers: SVM, RF, KNN, LR, DT, CNN, Gradient Boosting Decision Tree (GBDT), Adaptive Boosting (Adaboost), GaussianNB (GNB), Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), where the first six algorithms were used by the testing methods listed in Table 3. In this experiment, we train the individual classification models using the same training dataset and then use our anomaly detector to select the top-5 predictions (Section 3.4) to vote for the outcome. As can be seen from Figure 9, an ensemble learning-based approach can improve the sensing performance over using a single monolithic model. This observation is in line with prior studies [96]. However, a simple ensemble approach is inadequate for addressing the challenges in a dynamic environment as poor model predictions can mislead the ensemble outcomes. By filtering out poor model predictions, Rise further boosts the ensemble performance by on average 13%. This shows that Rise and ensemble learning can be combined together to improve sensing robustness.

5.4 Individual Case Studies

5.4.1 Case study 1. In this experiment, we test two WiFi-based systems: WiG [33] and WiAG [71], two RFID-based systems: TACT [80], AllSee [40], and one Ultrasound-based system: EI [83]. The experiments were conducted in the controlled environment using a robotic arm (Figure 3a). We change where the gesture is performed and adding a chair to the sensing area to mimic environmental changes. This results in five scenes ($S1-S5$) as depicted earlier in Figure 1a. WiAG is a data augmentation approach that uses transfer functions to generate virtual samples across different gesture locations (but cannot deal with the scenarios of placing a chair - S4 and S5). For this method, we also follow the evaluation setup described in [71] by

[4]The standard formula for computing the false positive rate is $FP/(FP+TN)$, where FP and TN are the number of false-positive and true-negative samples. This is different from how precision is computed.

Case Study 1

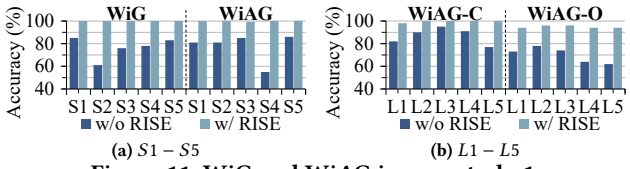
True label		RISE Predictions																															
		WiG					WiAG					WiAG-C					WiAG-O					TACT					AllSee					EI				
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	L1	L2	L3	L4	L5	L1	L2	L3	L4	L5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
Drifting	■✓	23	58	42	38	30	35	34	26	81	25	25	17	7	16	40	41	37	41	53	60	30	68	26	23	33	58	18	36	38	46	23	81	29	28	68
	▲✓	0	4	0	0	0	1	0	0	2	0	2	0	1	0	1	5	1	3	9	3	1	4	3	3	7	0	0	0	6	4	1	4	2	2	7
Normal	■	4	12	1	2	0	0	0	1	0	1	6	0	1	0	0	5	2	5	10	6	1	2	3	3	1	12	0	0	5	6	0	2	0	1	2
	▲✓	153	106	137	140	150	144	146	153	97	154	138	157	165	158	133	120	134	125	102	105	148	106	208	151	139	110	162	144	131	124	156	93	149	149	103

Case Study 2

True label		RISE Predictions																															
		WiWho					WifiU					VibWrite-R					VibWrite-A					Taprint-R					Taprint-A					UDO-Free				M-Touch			
		S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	N	1	2	3	4	N	1	2	3	4	S	D	M	W	S	D	M	W	RP	LP	DC	DB	S	DP	T	M
Drifting	■✓	153	128	123	147	120	110	85	53	66	56	22	84	73	23	18	36	58	63	61	67	72	77	88	79	40	34	41	32	44	113	106	36	31	28	37	36
	▲✓	4	3	12	7	7	1	12	6	8	6	2	19	0	0	0	6	0	1	1	11	7	13	12	6	3	1	3	2	0	6	9	3	4	2	1	3
Normal	■	9	5	7	9	2	7	14	10	9	10	4	8	0	0	0	0	7	5	11	8	1	14	19	12	4	6	6	4	0	13	6	8	5	4	5	5
	▲✓	164	194	188	167	201	212	219	261	247	258	172	89	127	177	182	262	220	235	247	234	386	424	411	411	297	318	290	303	621	387	396	470	203	213	196	210

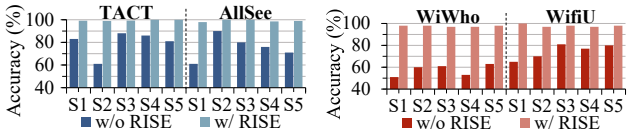
Case Study 3

Figure 10: Confusion matrices for predicting if a sample is drifting. Each row shows the number of samples of the true label and each column shows the predictions given by Rise for each evaluation setting.



(a) $S1 - S5$ (b) $L1 - L5$

Figure 11: WiG and WiAG in case study 1.



(a) Case study 1 (b) Case study 2

Figure 12: TACT and AllSee in case study 1 (a) and WiWho and WifiU in case study 2 (b).

performing the gesture in five other locations ($L1 - L5$ in Figure 3d) in addition to $S1 - S5$. This additional experiment was conducted in both the controlled (Figure 3e) and the office (Figure 4a) environments, involving 6 volunteers. We train the sensing model using leave-one-out cross-validation by training the model using data collected from 4 scenes and then test the trained model by mixing data collected from the remaining scene (see also Section 4.6.1).

Figures 11a - 13a show that minor changes in the environment can have a detrimental effect on the sensing performance. For example, WiAG gives an accuracy of around 55% when testing in S4 when its transfer function fails, and WiG gives an accuracy of 61% when testing in S2. We also observe a similar trend for RFID and Ultrasound based methods. Rise identifies at least 83% (Recall) of mispredictions. By using up to 3 labeled drifting samples to update the underlying model, Rise boosts the performance of the deployed model at least 98% of its static performance. As can be seen from the confusion matrix in Figure 10, Rise rarely miss-predicts the test samples, giving an average precision of 95%.

Compared to data augmentation techniques. Figure 11b shows while WiAG's data augmentation strategy works well in specific settings (e.g., L3) within the controlled environment, it can still suffer from significant performance degradation in many other settings. For example, its accuracy drops to 62% when testing in L5 in the office environment. In this experiment, Rise can identify at least 81% of the drifting samples (Recall), improving the robustness of WiAG in both the controlled and office environments. Using 3 labelled drifting samples to update the underlying model, Rise further boosts WiAG's performance to over 92%. From Figure 10, we see that Rise is also accurate in detecting drifting and normal samples, giving an average precision of 94%.

5.4.2 Case study 2. This experiment applies Rise to WiWho [93] and WifiU [81] for gait recognition. The experiment was conducted in the controlled environment to allow a better control of evaluation parameters. To introduce changes, we asked each of the 15 volunteers to walk along different paths, and placed a chair to the sensing area to simulate the change of the multi-path. This results in five scenes ($S1 - S5$). Figure 12b shows the performance of both systems decreases with environmental changes. WifiU exhibits more reliable performance than WiWho because it largely relies on environment-agnostic features like walking speed and step duration, showing that feature design is also important. As can be seen from the confusion matrix in Figure 10, Rise is effective in detecting drifting samples for this case study, giving an average precision of 93%. By labeling at most 3 drifting samples with incremental learning, Rise increases the sensing accuracy of both systems to above 97%.

5.4.3 Case study 3. We apply Rise to VibWrite [48] and Taprint [84] for finger input recognition and user identification, UDO-Free [69] for activity recognition, and M-Touch [64] for user identification. For VibWrite, we place a glass of water at four different locations while participants tapped on nine grid points (i.e., 1 – 4 in Figure 4b). We also use symbol N to denote the experimental setting where nothing was placed. For Taprint, we asked the participant to tap their 12 knuckles with a dry hand using (i) a standard force and speed (S) and (ii) different forces (D), and a wet hand (W) with a steady force and speed. Also, we asked the participant to tap their knuckles with a dry hand using a steady force and speed while walking in an outdoor setting (marked as M). For UDO-Free, we asked each volunteer to place the mobile phone in their right pocket (RP) and left pocket (LP) to perform a set of actions supported by UDO-Free. In addition to changing the location of the smartphone, we asked the participants to place the phone in the right pocket to (i) sit on chairs (DC) with heights between $20cm$ and $40cm$ (to perform the "Sitting" action), and (ii) ride a mountain and a road bike (DB) to perform the "Biking" action. For M-Touch, we asked each volunteer to perform the action in sitting (marked as S), standing (marked as DP), and walking (marked as M) postures while holding the smartphone in hands. Also, each volunteer was asked to perform the action in a sitting posture while the phone was placed on the table (marked as T). Like case study 1 (Section 5.4.1), we train the sensing model using leave-one-out cross-validation.

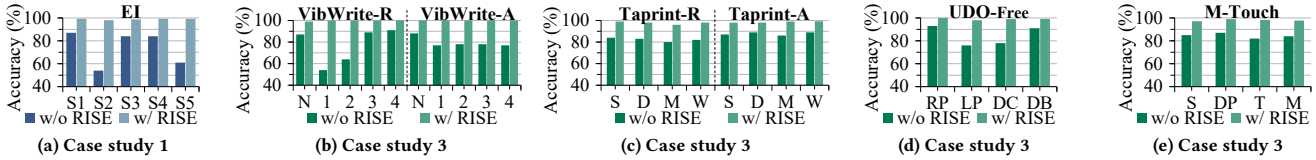


Figure 13: Sensing performance for EI in case study 1 (a) and methods in case study 3 (b, c, d, e).

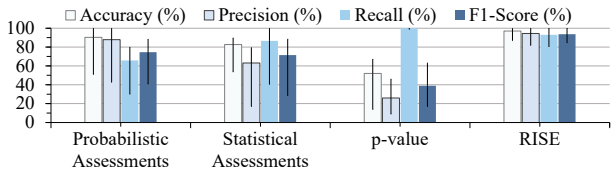


Figure 14: Detecting drifting samples when using different measures for anomaly detection.

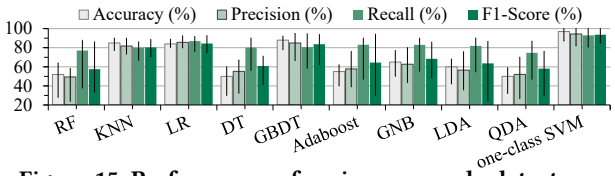


Figure 15: Performance of various anomaly detectors.

Figures 13b to 13e give the result of this experiment. For finger input recognition, putting a glass of water in positions 1 and 2 of Figure 4b cause the accuracy of VibWrite to drop to around 60%. By contrast, the tapping force and hand conditions have less impact on the accuracy of Taprint because the changes introduce negligible changes to the model inputs. However, Taprint's performance also drops from 99% (Table 3) in a static environment to 78% when the input conditions changed. From Figures 13d and 13e, we see UDO-Free and M-Touch are also sensitive to environmental change, where the accuracy of UDO-Free can drop from 98.8% (Table 3) to be less than 80% when it is tested in the LP and DC setting. For this case study, Rise successfully detects an average of 90% (Recall) of the drifting samples. Like other case studies, by using up to 3 drifting samples to retrain the sensing model, we can bring its performance back to above 96%.

5.5 Model Analysis

5.5.1 Measures for anomaly detection. Figure 14 shows performance for detecting drifting samples when only using the probability (Section 3.1) or statistical vector (Section 3.2) over Rise. We also consider using the p-value of the sensing model output, which is a standard approach for using CP in detecting drifting samples [39]. The results are averaged across our case studies and evaluation setups, where the min-max bar shows the variance across settings.

By using both the probability and the statistical vectors for anomaly detection, Rise gives the best overall results across the metrics. While using the p-value alone gives a high Recall (i.e., CP identifies most of the drifting samples), this strategy gives a low precision, meaning that a large number of normal samples are mislabeled as drifting. Using only the probability vector leads to a low Recall, indicating this strategy often miss drifting samples. Similarly, using only the statistical vector gives a low precision because it frequently mislabels normal samples. By combining probabilistic and statistical assessments, Rise gives the best overall performance by achieving 92% or above for the three evaluation metrics. We note that Rise is the first work in combining probabilistic and statistical assessments and anomaly detection.

5.5.2 Choices of anomaly detectors. Figure 15 shows the performance for using different models for anomaly detection (by taking the probability and statistical vector as the model input). All the models were built from the same training dataset used by the underlying sensing model. Here, we report the average performance across our case studies and evaluation setups and show the variance as the min-max bar. This diagram shows that the one-class SVM used by Rise gives the best performance across evaluation metrics.

5.5.3 Runtime overhead. The runtime overhead of Rise mainly comes from computing the nonconformity score and performing anomaly detection. Since the nonconformity score of the calibration set was computed off-line, it is a *one-off* cost. The time for measuring the confidence and credibility score of the test sample is small, less than 20 milliseconds (ms), using an unoptimized, single-threaded Python script running on a laptop CPU. The overhead of the anomaly detection is also negligible, less than 5 ms during our evaluation. The runtime overhead of Rise can be further reduced by using a statically compiled language with parallelization.

6 DISCUSSIONS

Generalization. Rise can be applied to any probability-based classifier using the generic nonconformity function presented in Table 1. Rise aims to retain the performance of a learning-based system in the presence of environmental changes. We have shown that a effective model in the original training-test dataset can still suffer from changes where Rise will be helpful. CP also supports regression methods so Rise can be extended to regression-based models. Rise currently does not work with model-based sensing because it uses the probability distribution given by the target model to compute confidence and credibility. We see it as an exciting challenge to evaluate and validate the assumptions an analytic method builds upon. Can we have a generic approach to test if the assumptions a model-based approach relies on are still valid?

Combining with other methods. Rise is not designed to replace existing techniques on improving sensing robustness at the design phase [71, 83, 91, 96]. Instead, Rise provides a way to detect ageing models during deployment and hence is orthogonal to existing efforts in the area. For example, Rise can use to detect if adding additional training samples can offer new information to improve the sensing model performance. This allows us to reduce the cost of data labeling when learning a transfer function [96]. It can also be used to detect if the performance of a deployed model starts to deteriorate and additional remediation should be used.

Human verification. Rise is also not designed to completely eliminate human intervention during a deployment. Instead, it offers a

new way to detect data drift and ageing sensing models after deployment. Our work is useful because it only asks for user verification when data drifting is likely to occur. The user intervention can be of different forms, such as asking for the target user to confirm if the system's prediction is correct or to the correct label or using a second verification method (such as entering a pin or switching to a vision-based method [15]) to verify the user identity. The user feedback can then be added back to the training dataset to retrain the sensing model using our training APIs. We would like to remark that user intervention and verification are only possible once data drift is detected, which is the focus of this research.

Rise robustness. We believe Rise is robust when it is used with incremental learning. Our anomaly detector is an unsupervised learning method that does not rely on the iid assumption. Our CP model is updated whenever drifting samples are used to retrain the underlying model. Therefore, Rise not only helps to adapt the wireless sensing, but can also adapt its own performance to changes in the underlying signal environment.

Model interpretability. One way for gaining insight into the root cause of a misprediction is to train an interpretable, surrogate model to approximate the predictions of the underlying model [58]. A key challenge to do so for wireless sensing is that, unlike image and text inputs, the original wireless signals are not explicitly correlated with high-level semantics. Given the wide use of machine-learning models in wireless sensing, how to link the signals with suitable semantics is an interesting research challenge.

7 RELATED WORK

Machine learning is widely used to support wireless sensing tasks. This technique learns how to map the wireless signal characteristics to the target activity based on empirical observations. A learning-based approach has the advantage of being portable across application domains as machine learning has no a *priori* assumption about the sensing task [11, 12, 23, 40, 43, 45, 49, 62, 64, 71, 78, 82, 83, 90, 91, 97, 100]. Unfortunately, machine learning is known to be brittle to changes. Studies show that changes like moving or adding furniture can have a significantly negative impact on the performance of a learning-based system [64, 71, 74, 81]. This issue also manifests across wireless signals and sensing tasks [38, 59].

An alternative approach is to develop environment-agnostic methods – often with the help of parametric models or the specialized hardware – to cancel the changes in the spatial domain of the environment. Examples of such systems include sensing systems build on FMCW [3, 52, 79, 92] and physiological data [17, 89]. These approaches can generalize to different environments and handle multi-path. However, these solutions often require expert involvement to construct a sensing model for each application domain and do not scale to a diverse set of tasks.

Efforts have been made to improve the robustness of a learning-based sensing system. One solution is to produce virtual samples to improve the coverage of the training data [71, 96]. Others leverage the common knowledge extracted from multiple sensing tasks to enhance the generalization ability of the sensing model [28], or designing wireless signal features that are intrinsically more resilient to environment changes [83, 91]. Some most recent works suggested that neural networks could be more resilient to changes as the latent feature space better generalizes to new variants [42, 75]. All of these attempts focus on the design and training stage of a sensing system, but do not address the challenge of adapting to changes after model deployment. Often, many of the changes during a deployment are hard to anticipate at the design time.

Rise offers a complementary approach to enhance learning-based methods during deployment time. It achieves this by identifying test inputs that are likely to be mispredicted by a sensing model. Providing this capability allows the deployed system to adapt to changes by updating the model using incremental retraining or online learning [4, 25]. It also improves sensing robustness by rejecting mispredictions. As we have shown in the paper, Rise can be easily integrated with incremental and ensemble learning [18, 29, 35, 96] to improve sensing performance, while minimizing human involvement and the associated cost.

Rise is the first work in employing CP for wireless sensing and showing how CP can be combined with incremental and ensemble learning to improve sensing performance. Our work is inspired by recent studies for using CP in malware detection [39, 55], anomaly detection [87], and drug discovery [19]. Unlike Rise, all these recent works use a single scalar value – the *p-value* – for decision making. Our novel combination of probabilistic and statistical assessment vectors and anomaly detection gives better performance.

Some most recent works in deep neural networks estimate the prediction uncertainties by applying a distance metric to the network activation [30] or using a kernel function to measure the probability density of each network layer [86]. While promising, these techniques are specific to predictive models built upon artificial neural networks.

8 CONCLUSIONS

We have presented Rise, a generic framework for identifying and mitigating performance degradation issues when deploying a learning-based wireless sensing system in a changing environment. We have empirically demonstrated that a machine-learning model that is highly effective in the original training-test dataset can still suffer from small environmental changes during a deployment. To retain the performance of a learning-based system in the presence of environmental changes, Rise uses probabilistic and statistical assessments to filter out unreliable predictions and to support the continues improvement of a deployed model.

We evaluate Rise by applying it to 11 representative sensing systems. Our extensive evaluation shows that Rise is effective in detecting unreliable classification decisions by successfully identifying on average 92.3% of the mispredictions. The diversity of case studies and the promising experimental results provide compelling evidence in favor of Rise being generalizable. We demonstrate that Rise can be used together with incremental and ensemble learning to continuously improve the performance of a deployed sensing model. The result is a new way for improving wireless sensing robustness and performance during a deployment.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) under grant agreements 61972314, 61872294 and 62061146001, the International Cooperation Project of Shaanxi Province under grant agreements 2020KWZ-013, 2021KW-15, and 2021KW-04.

REFERENCES

[1] Adams, N. Dataset shift in machine learning. *Journal of the Royal Statistical Society* (2010).

[2] Ade, R., and Deshmukh, P. Methods for incremental learning: a survey. *International Journal of Data Mining & Knowledge Management Process* (2013).

[3] Adib, F., Kabelac, Z., and Katabi, D. Multi-person localization via {RF} body reflections. In *NSDI* (2015).

[4] Anderson, T. *The theory and practice of online learning*. Athabasca University Press, 2008.

[5] Appel-Hansen, and J. Reflectivity level of radio anechoic chambers. *IEEE Transactions on Antennas & Propagation* (2003).

[6] Ayyalasomayajula, R. S., Arun, A., Wu, C., Sharma, S., Sethi, A. R., Vasisht, D., and Bharadia, D. Deep learning based wireless localization for indoor navigation. In *MobiCom* (2020).

[7] Balasubramanian, V., Ho, S.-S., and Vovk, V. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

[8] Balasubramanian, V. N., Baker, A., Yanez, M., Chakraborty, S., and Panchanathan, S. Pycp: an open-source conformal predictions toolkit. In *IFIPAICT* (2013).

[9] Barbero, F., Pendlebury, F., Pierazzi, F., and Cavallaro, L. Transcending transcend: Revisiting malware classification with conformal evaluation. *arXiv* (2020).

[10] Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

[11] Bo, C., Qian, Z., Zhao, R., Dong, L., and Dong, W. Sgrs: A sequential gesture recognition system using cots rfid. In *WCNC* (2018).

[12] Bo, W., Wen, H., Yang, M., and Chou, C. T. From real to complex: Enhancing radio-based activity recognition using complex-valued csi. *ACM Transactions on Sensor Networks* (2018).

[13] Browne, M. W. Cross-validation methods. *Journal of mathematical psychology* (2000).

[14] Bui, N., Pham, N., Barnitz, J. J., Zou, Z., Nguyen, P., Truong, H., Kim, T., Farrow, N., Nguyen, A., Xiao, J., et al. ebp: A wearable system for frequent and comfortable blood pressure monitoring from user's ear. In *MobiCom* (2019).

[15] Cai, H., Korany, B., Karanam, C. R., and Mostofi, Y. Teaching rf to sense without rf training measurements. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2020).

[16] Chen, B., Li, H., Li, Z., Chen, X., Xu, C., and Xu, W. Thermowave: a new paradigm of wireless passive temperature monitoring via mmwave sensing. In *MobiCom* (2020).

[17] Chen, Y., Yang, Z., Abbou, R., Lopes, P., Zhao, B. Y., and Zheng, H. User authentication via electrical muscle stimulation. In *CHI* (2021).

[18] Chen, Z., Jiang, C., and Xie, L. A novel ensemble elm for human activity recognition using smartphone sensors. *IEEE Transactions on Industrial Informatics* (2018).

[19] Cortés-Ciriano, I., and Bender, A. Deep confidence: a computationally efficient framework for calculating reliable prediction errors for deep neural networks. *Journal of chemical information and modeling* (2018).

[20] Devetyarov, D., and Nouretdinov, I. Prediction with confidence based on a random forest classifier. In *IFIPAICT* (2010).

[21] Ding, J., and Chandra, R. Strobe – towards low cost soil sensing using wi-fi. In *MobiCom* (2019).

[22] Doost-Mohammady, R., Bejarano, O., and Sabharwal, A. Good times for wireless research. In *WiNTECH* (2020).

[23] Fan, X., Wei, G., and Jiangchuan, L. Tagfree activity identification with rfids. *IMWUT* (2018).

[24] Farrukh, H., Aburas, R. M., Cao, S., and Wang, H. Facerevelio: a face liveness detection system for smartphones with a single front camera. In *MobiCom* (2020).

[25] Förster, K., Biasiucci, A., Chavarriaga, R., Millán, J. d. R., Roggen, D., and Tröster, G. On the use of brain decoded signals for online user adaptive gesture recognition systems. In *Pervasive* (2010).

[26] Gepperth, A., and Hammer, B. Incremental learning algorithms and applications. In *ESANN* (2016).

[27] Gong, T., Cho, H., Lee, B., and Lee, S.-J. Knocker: Vibroacoustic-based object recognition with smartphones. *IMWUT* (2019).

[28] Gong, T., Kim, Y., Shin, J., and Lee, S.-J. Metasense: Few-shot adaptation to untrained conditions in deep mobile sensing. In *SenSys* (2019).

[29] Guan, Y., Li, C.-T., and Roli, F. On reducing the effect of covariate factors in gait recognition: a classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence* (2014).

[30] Guerriero, A., Pietrantuono, R., and Russo, S. Operation is the hardest teacher: estimating dnn accuracy looking for mispredictions. In *ICSE* (2021), IEEE.

[31] Halperin, D., Hu, W., Sheth, A., and Wetherall, D. Tool release: Gathering 802.11n traces with channel state information. In *SIGCOMM* (2011).

[32] Han, C., Wu, K., Wang, Y., and Ni, L. M. Wifall: Device-free fall detection by wireless networks. In *INFOCOM* (2014).

[33] He, W., Wu, K., Zou, Y., and Zhong, M. Wig: Wifi-based gesture recognition system. In *ICCCN* (2015).

[34] Hodge, V., and Austin, J. A survey of outlier detection methodologies. *Artificial intelligence review* (2004).

[35] Hu, C., Chen, Y., Hu, L., and Peng, X. A novel random forests based class incremental learning method for activity recognition. *Pattern Recognition* (2018).

[36] Huang, H., and Lin, S. Met: a magneto-inductive sensing based electric toothbrushing monitoring system. In *MobiCom* (2020).

[37] Jiang, W., Xue, H., Miao, C., Wang, S., Lin, S., Tian, C., Murali, S., Hu, H., Sun, Z., and Su, L. Towards 3d human pose construction using wifi. In *MobiCom* (2020).

[38] Jie, W., Gao, Q., Miao, P., and Fang, Y. Device-free wireless sensing: Challenges, opportunities, and applications. *IEEE Network* (2018).

[39] Jordaney, R., Sharad, K., Santanu, K. D., Wang, Z., Papini, D., Nouretdinov, I., and Cavallaro, L. Transcend: Detecting concept drift in malware classification models. In *USENIX Security* (2017).

[40] Kellogg, B., Talla, V., and Gollakota, S. Bringing gesture recognition to all devices. In *NSDI* (2014).

[41] Korany, B., Karanam, C. R., Cai, H., and Mostofi, Y. XModal-ID: Using WiFi for Through-Wall Person sIdentification from Candidate Video Footage. In *MobiCom* (2019).

[42] Lee, W., Kim, M., and Cho, D.-H. Deep cooperative sensing: Cooperative spectrum sensing based on convolutional neural networks. *IEEE Transactions on Vehicular Technology* (2019).

[43] Li, S., Ashok, A., Zhang, Y., Xu, C., and Gruteser, M. Whose move is it anyway? authenticating smart wearable devices using unique head movement patterns. In *PerCom* (2016).

[44] Li, T., Bai, D., Prioleau, T., Bui, N., Vu, T., and Zhou, X. Noninvasive glucose monitoring using polarized light. In *SenSys* (2020).

[45] Liu, C., Xiong, J., Cai, L., Feng, L., Chen, X., and Fang, D. Beyond respiration: Contactless sleep sound-activity recognition using RF signals. *IMWUT* (2019).

[46] Liu, J., Liu, H., Chen, Y., Wang, Y., and Wang, C. Wireless sensing for human activity: A survey. *IEEE Communications Surveys & Tutorials* (2019).

[47] Liu, J., Shi, C., Chen, Y., Liu, H., and Gruteser, M. Cardiocam: Leveraging camera on mobile devices to verify users while their heart is pumping. In *MobiSys* (2019).

[48] Liu, J., Wang, C., Chen, Y., and Saxena, N. Vibwrite: Towards finger-input authentication on ubiquitous surfaces via physical vibration. In *CCS* (2017).

[49] Lu, L., Yu, J., Chen, Y., Liu, H., Zhu, Y., Kong, L., and Li, M. Lip reading-based user authentication through acoustic sensing on smartphones. *IEEE/ACM Transactions on Networking* (2019).

[50] Ma, D., Lan, G., Hassan, M., Hu, W., Upama, M. B., Uddin, A., and Youssef, M. Solargest: Ubiquitous and battery-free gesture recognition using solar cells. In *MobiCom* (2019).

[51] Ma, Y., Zhou, G., and Wang, S. Wifi sensing with channel state information: A survey. *ACM Computing Surveys* (2019).

[52] Mao, W., He, J., and Qiu, L. Cat: high-precision acoustic motion tracking. In *MobiCom* (2016).

[53] Marco, V. S., Taylor, B., Porter, B., and Wang, Z. Improving spark application throughput via memory aware task co-location: A mixture of experts approach. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference* (2017).

[54] Mascolo, C. Mobile health diagnostics through audio signals. In *HealthDL* (2020).

[55] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., and Cavallaro, L. Tesseract: Eliminating experimental bias in malware classification across space and time. In *USENIX Security* (2019).

[56] Pradhan, S., Baig, G., Mao, W., Qiu, L., Chen, G., and Yang, B. Smartphone-based acoustic indoor space mapping. *IMWUT* (2018).

[57] Pu, Q., Jiang, S., and Gollakota, S. Whole-home gesture recognition using wireless signals. In *SIGCOMM* (2013).

[58] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?" explaining the predictions of any classifier. In *KDD* (2016).

[59] Savazzi, S., Sigg, S., Nicoli, M., Rampa, V., Kianoush, S., and Spagnolini, U. Wireless sensing for device-free recognition of human motion. *Radar for Indoor Monitoring: Detection, Classification, and Assessment* (2017).

[60] Schölkopf, B., Platt, J. C., et al. Support vector method for novelty detection. In *NIPS* (1999).

[61] Shafer, G., and Vovk, V. A tutorial on conformal prediction. *Journal of Machine Learning Research* (2007).

[62] Shangguan, L., Zhou, Z., and Jamieson, K. Enabling gesture-based interactions with objects. In *MobiSys* (2017).

[63] Song, X., Yang, B., Yang, G., Chen, R., Forno, E., Chen, W., and Gao, W. Spirosonic: monitoring human lung function via acoustic sensing on commodity smartphones. In *MobiCom* (2020).

[64] Song, Y., Cai, Z., and Zhang, Z. L. Multi-touch authentication using hand geometry and behavioral information. In *S&P* (2017).

[65] Swami, A., and Jain, R. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* (2012).

[66] Tang, W., and Sazonov, E. S. Highly accurate recognition of human postures

and activities through classification with rejection. *IEEE journal of biomedical and health informatics* (2014).
[67] Tax, D. M. J. One-class classification: Concept learning in the absence of counter-examples. *Delft University of Technology* (2001).
[68] Tsymbal, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* (2004).
[69] Ustev, Y. E., Incel, O. D., and Ersoy, C. User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal. In *UbiComp* (2013).
[70] Van der Maaten, L., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research* (2008).
[71] Virmani, A., and Shahzad, M. Position and orientation agnostic gesture recognition using wifi. In *MobiSys* (2017).
[72] Vovk, V., Gammerman, A., and Shafer, G. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.
[73] Wang, A., Sunshine, J. E., and Gollakota, S. Contactless infant monitoring using white noise. In *MobiCom* (2019).
[74] Wang, J., Chang, L., Abari, O., and Keshav, S. Are rfid sensing systems ready for the real world? In *MobiSys* (2019).
[75] Wang, J., Gao, Q., Ma, X., Zhao, Y., and Fang, Y. Learning to sense: Deep learning for wireless sensing with less training efforts. *IEEE Wireless Communications* (2020).
[76] Wang, J., Gao, Q., Pan, M., and Fang, Y. Device-free wireless sensing: Challenges, opportunities, and applications. *IEEE Network* (2018).
[77] Wang, J., Li, J., Mazaheri, M. H., Katsuragawa, K., Vogel, D., and Abari, O. Sensing finger input using an rfid transmission line. In *SenSys* (2020).
[78] Wang, J., Xiong, J., Chen, X., Jiang, H., and Fang, D. Tagscan: Simultaneous target imaging and material identification with commodity rfid devices. In *MobiCom* (2017).
[79] Wang, T., Zhang, D., Zheng, Y., Gu, T., Zhou, X., and Dorizzi, B. C-fmcw based contactless respiration detection using acoustic signal. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2018).
[80] Wang, Y., and Zheng, Y. Modeling rfid signal reflection for contact-free activity recognition. *IMWUT* (2019).
[81] Wei, W., Liu, A. X., and Shahzad, M. Gait recognition using wifi signals. In *UbiComp* (2016).
[82] Wei, W., Liu, A. X., Shahzad, M., Kang, L., and Lu, S. Understanding and modeling of wifi signal based human activity recognition. In *MobiCom* (2015).
[83] Wenjun, J., Chenglin, M., Fenglong, M., Shuochao, Y., Yaqing, W., Ye, Y., Hongfei, X., Chen, S., Xin, M., Dimitrios, K., Wenyao, X., and Lu, S. Towards environment independent device free human activity recognition. In *MobiCom* (2018).
[84] Wenqiang, C., Lin, C., Yandao, H., Xinyu, Z., Lu, W., and Kaishun, W. Taprint: Secure text input for commodity smart wristbands. In *MobiCom* (2019).
[85] Xiao, C., Han, D., Ma, Y., and Qin, Z. Csigan: Robust channel state information-based activity recognition with gans. *IEEE Internet of Things Journal* (2019).
[86] Xiao, Y., Beschastnikh, I., Rosenblum, D. S., Sun, C., Elbaum, S., Lin, Y., and Dong, J. S. Self-checking deep neural networks in deployment. In *ICSE* (2021), IEEE.
[87] Xie, X., Jin, Z., Wang, J., Yang, L., Lu, Y., and Li, T. Confidence guided anomaly detection model for anti-concept drift in dynamic logs. *Journal of Network and Computer Applications* (2020).
[88] Xie, Y., Xiong, J., and Jamieson, K. md-track: Leveraging multi-dimensionality for passive indoor wi-fi tracking. In *MobiCom* (2019).
[89] Xu, X., Yu, J., Chen, Y., Hua, Q., Zhu, Y., Chen, Y.-C., and Li, M. Touchpass: towards behavior-irrelevant on-touch user authentication on smartphones leveraging vibrations. In *MobiCom* (2020).
[90] Yu, Y., Wang, D., Zhao, R., and Zhang, Q. Rfid based real-time recognition of ongoing gesture with adversarial learning. In *SenSys* (2019).
[91] Yue, Z., Yi, Z., Kun, Q., Guidong, Z., Yunhao, L., Chenshu, W., and Zheng, Y. Zero-effort cross-domain gesture recognition with wi-fi. In *MobiSys* (2019).
[92] Yun, S., Chen, Y.-C., and Qiu, L. Turning a mobile device into a mouse in the air. In *MobiSys* (2015).
[93] Zeng, Y., Pathak, P. H., and Mohapatra, P. Wiwho: Wifi-based person identification in smart spaces. In *IPSN* (2016).
[94] Zhang, D., Wang, J., Jang, J., Zhang, J., and Kumar, S. On the feasibility of wi-fi based material sensing. In *MobiCom* (2019).
[95] Zhang, F., Niu, K., Xiong, J., Jin, B., Gu, T., Jiang, Y., and Zhang, D. Towards a diffraction-based sensing approach on human activity recognition. *IMWUT* (2019).
[96] Zhang, J., Tang, Z., Li, M., Fang, D., Nurmi, P., and Wang, Z. Crosssense: Towards cross-site and large-scale wifi sensing. In *MobiCom* (2018).
[97] Zhang, O., and Srinivasan, K. Mudra: User-friendly fine-grained gesture recognition using wifi signals. In *CoNEXT* (2016).
[98] Zhang, X., Li, W., Chen, X., and Lu, S. Moodexplorer: Towards compound emotion detection via smartphone sensing. *IMWUT* (2018).
[99] Zhao, M., Adib, F., and Katabi, D. Emotion recognition using wireless signals. In *MobiCom* (2016).
[100] Zhao, M., Tian, Y., Zhao, H., Alsheikh, M., Li, T., Hristov, R., Kabelac, Z., Katabi, D., and Torralba, A. Rf-based 3d skeletons. In *SIGCOMM* (2018).
[101] Žliobaitė, I. Learning under concept drift: an overview. *Computer Science* (2010).