Towards Receiver-Agnostic and Collaborative Radio Frequency Fingerprint Identification

Guanxiong Shen, Junqing Zhang, *Member, IEEE*, Alan Marshall, *Senior Member, IEEE*, Roger Woods, *Senior Member, IEEE*, Joseph Cavallaro, *Fellow, IEEE*, and Liquan Chen, *Senior Member, IEEE*

Abstract—Radio frequency fingerprint identification (RFFI) is an emerging device authentication technique, which exploits the hardware characteristics of the RF front-end as device identifiers. The receiver hardware impairments interfere with the feature extraction of transmitter impairments, but their effect and mitigation have not been comprehensively studied. In this paper, we propose a receiver-agnostic RFFI system by employing adversarial training to learn the receiver-independent features. Moreover, when there are multiple receivers, collaborative inference are designed to enhance classification accuracy. Finally, we show how it is possible to leverage fine-tuning for further improvement with fewer collected signals. To validate the approach, we have conducted extensive experimental evaluation by applying the approach to a LoRaWAN case study involving ten LoRa devices and 20 software-defined radio (SDR) receivers. The results show that receiver-agnostic training enables the trained neural network to become robust to changes in receiver characteristics. The collaborative inference improves classification accuracy by up to 20% beyond a single-receiver RFFI system and fine-tuning can bring a 40% improvement for underperforming receivers. The system is further evaluated on a more practical testbed. By making additional use of online augmentation and multi-packet inference, the identification accuracy is improved from 50% to 90% at 10 dB.

Index Terms—Internet of Things, LoRa/LoRaWAN, device authentication, radio frequency fingerprint, adversarial training, collaborative inference

1 Introduction

The number of Internet of things (IoT) devices has exploded in recent years with the emergence of different standards such as Bluetooth low energy (BLE), WiFi, LoRa, Sigfox, etc. The authentication of IoT devices is critical for ensuring that received messages are sent from authorized and legitimate devices [1]. Conventional device authentication schemes usually rely on cryptographic solutions and use software addresses (e.g. MAC address) as device identifiers, but they are highly susceptible to tampering, resulting in possible spoof attacks [2]. Cryptographic solutions include public key-based authentication and symmetric key-based authentication. For the former, public-key cryptography is quite complex and its implementation might be beyond the resources affordable in IoT devices. Indeed, it is quite challenging to securely and efficiently establish symmetric keys in IoT [3].

Radio frequency fingerprint identification (RFFI) is a promising non-cryptographic device authentication technique. Like human fingerprints, wireless devices have unique radio frequency fingerprints (RFFs) resulting from the impairments of the hardware components in the transmitter front-end which are usually challenging to mimic. The impairments generally include oscillator drift, mixer imperfection, power amplifier nonlinearity, etc. These have been modeled in previous studies [4], [5] and are deemed to be suitable for device identification. The RFFI system extracts unique features from wireless signals transmitted by IoT devices to infer their identities. The RFFI can be formulated as a multi-class classification problem, and there are numerous examples of where deep learning has been used to boost RFFI performance [5]–[31].

There are several key challenges with RFFI development. Firstly, the captured signals are not only distorted by the transmitter hardware impairments but also by the wireless channel [7], [11], [13], [22]. To address this, researchers design channel-agnostic signal representations or use data augmentation techniques for mitigation [7], [11], [22]. In addition, the transmitter hardware characteristics can change

Manuscript received xxx; revised xxx; accepted xxx. Date of publication xxx; date of current version xxx. The work was in part supported by National Key Research and Development Program of China under grant ID 2020YFE0200600. The work of J. Zhang was also in part supported by UK Engineering and Physical Sciences Research Council under grant ID EP/V027697/1 and UK Royal Society Research Grants under grant IDs RGS/R1/191241 and RGS/R1/231435. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising. The review of this paper was coordinated by xxx. (Corresponding author: Junqing Zhang.)

- *G. Shen, J. Zhang, and A. Marshall are with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, L69 3GJ, United Kingdom. (email: Guanxiong.Shen@liverpool.ac.uk; junqing.zhang@liverpool.ac.uk; alan.marshall@liverpool.ac.uk)*
- *R. Woods is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, BT9 5AG, United Kingdom. (email: r.woods@qub.ac.uk)*
- *J. Cavallaro is with the Department of Electrical and Computer Engineering, Rice University, Houston, USA. (email: cavallar@rice.edu)*
- *L. Chen is with the School of Cyber Science and Engineering, Southeast University, Nanjing, 210096, China and also with the Purple Mountain Laboratories for Network and Communication Security, Nanjing, 211111, China. (e-mail: lqchen@seu.edu.cn)*

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.
Digital Object Identifier xxx

over time, thus degrading the RFFI performance. Previous studies indicate that the transmitter oscillator is the primary contributor to the characteristics drift, and frequency offset compensation is demonstrated to be an effective mitigation [9], [13].

Although tremendous efforts have been made to improve RFFI performance, the receiver effect is often overlooked in previous studies. The changes in receiver hardware characteristics can seriously affect RFFI performance. Most existing RFFI work assumes that the same receiver is used during training and inference and that the receiver characteristics do not change over time [7]–[10], [22], [26]. However, this assumption does not always hold in practical IoT applications. For instance, mobile IoT devices will be served by different access points/gateways, depending on their coverage. Furthermore, even if the same receiver is used, the hardware characteristics of low-cost receivers may vary over time. Although re-training on the new receiver is a possible solution, the long training time is often not affordable when there are numerous end nodes operating in the IoT network. Therefore, there is an urgent need for a receiver-agnostic RFFI system that can be deployed in a highly practical manner.

As wireless transmissions are broadcast and can be captured by any receivers within range, it is, therefore, possible to design a collaborative RFFI protocol that can enhance system performance. In IoT applications, multiple receivers can be present with numerous gateways in LoRaWAN and multiple access points in WiFi enterprise networks, but critically, to the best of our knowledge, there are only two papers that have explored using multiple receivers in RFFI systems [18], [32]. However, the algorithm in [32] is not applicable to deep learning-based RFFI systems, and the method proposed in [18] is only evaluated with limited experimental work.

In this paper, a receiver-agnostic and collaborative RFFI protocol is designed which is robust to receiver characteristic variations. In particular, multiple receivers were used to collect sufficient packets from devices under test (DUTs) in order to train a receiver-agnostic neural network. During the inference, receivers are equipped with the trained receiver-agnostic neural network model. Once a packet is captured, the receivers initially make independent inferences which are then fused to permit better classification performance. For our experimental evaluation, we used LoRa/LoRaWAN as a case study as it is a suitable technique to demonstrate the ideas. Specifically, we employed ten commercial-off-the-shelf (COTS) LoRa nodes as DUTs and 20 software-defined radio (SDR) platforms as the LoRa gateways. Whilst the work focuses on LoRa/LoRaWAN as a case study, our receiver-agnostic approach is applicable to any RFFI system and the collaborative protocol is suitable for any wireless technique with multiple receivers operating simultaneously. The detailed contributions of this work include:

- The receiver effects on RFFI are experimentally investigated. The RFFI system implemented on low-end SDR receivers (e.g., RTL-SDR) shows an accuracy drop of 40% over four continuous days, which is probably due to the unstable hardware characteristics. Moreover, we show that changing the receiver in an RFFI system can result in serious performance degradation. For example, the neural network trained with an RTL-SDR only achieves less than 20% accuracy when the test was using a USRP B200 SDR.
- A receiver-agnostic neural network for RFFI is proposed using adversarial training. We guide the neural network to learn receiver-independent features so that it is robust to performance degradation caused by receiver drift and change. We propose two training strategies for receiver-agnostic RFFI, namely homogeneous and heterogeneous training, depending on the diversity of the training receivers. The results show that the neural network trained with heterogeneous adversarial training achieves better performance than the homogeneous one. Its classification accuracy is over 75% for all of the 20 SDRs and even exceeds 95% on receivers other than RTL-SDRs. Compared to the conventional approach, receiver-agnostic training effectively prevents drastic performance degradation on previously unseen receivers.
- A collaborative RFFI system with soft or adaptive soft fusion schemes is proposed and experimentally evaluated in both residential and office building environments. All the receivers are equipped with the same receiver-agnostic neural network. They make independent inferences at the edge and upload them to a network server. The inferences are then fused by soft fusion or adaptive soft fusion schemes to achieve higher accuracy. The experimental results show that collaborative RFFI can improve the classification accuracy by up to 20% compared to the single-receiver RFFI system. When combining the online augmentation and multi-packet inference techniques proposed in our previous work [31], the system performance is enhanced by approximately 40% at 10 dB, i.e., from 50% to 90%.
- A further fine-tuning technique is proposed to mitigate receiver effects as even after receiver-agnostic training, the neural network still may not reach satisfactory accuracy on some unseen receivers. To address this, we propose a fine-tuning approach that can collect a few packets with the new receiver to slightly adjust the neural network parameters. The neural network achieves higher performance with fine-tuning because it better adapts to the new receiver by re-learning the characteristics of the received signal. Experimental results show a further accuracy improvement of up to 40% on receiver-agnostic neural networks.

It is worth noting that this paper focuses on the closed-set classification problem, i.e., the training and test devices are the same. The design methodology of openset RFFI protocols with rogue device detection function can be found in [11], [28]. The code and dataset will be released to the community upon formal acceptance of the paper.

The rest of the paper is organized as follows. Section 2 introduces conventional RFFI systems and defines the research challenge. Section 3 gives a system overview and elaborates on each of the system modules. Section 4 gives details of the LoRa/LoRaWAN case study. A controlled ex-

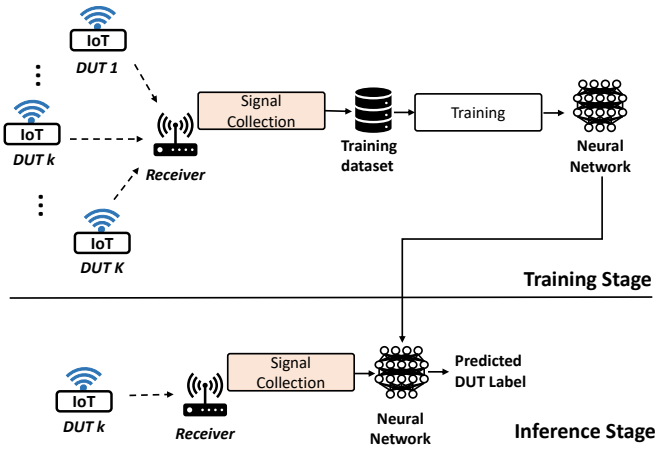


Fig. 1: Overview of a conventional deep learning-based RFFI system.

perimental evaluation of the receiver-agnostic training and collaborative RFFI system carried out in a residential room, is given in Section 5. Section 6 provides the experimental results in an office building in which the collaborative RFFI is further evaluated. Related work is provided in Section 7 and Section 8 concludes this paper.

2 Conventional Approach and Problem Statement

2.1 Conventional Deep Learning-based RFFI Approach

An RFFI system aims to classify K IoT end nodes, i.e. devices under test (DUTs), in a wireless network by analyzing the received physical layer signals, as shown in Fig. 1. The received baseband signal $y(t)$ can be mathematically given as

$$y(t) = \mathcal{G}\Big(h(t) * \mathcal{F}^k(x(t))\Big) + n(t), \tag{1}$$

where $\mathcal{G}(\cdot)$ denotes the hardware effects of the receiver, $h(t)$ is the wireless channel impulse response, $\mathcal{F}^k(\cdot)$ represents the transmitter chain effect of DUT k, $n(t)$ is the additive white Gaussian noise (AWGN) and $*$ denotes the convolution operation. The goal of an RFFI system is to predict the transmitter label k by analyzing the collected signal, $y(t)$. Deep learning algorithms are seen as a suitable solution due to their excellent feature extraction capabilities.

As shown in Fig. 1, a conventional deep learning-based RFFI system[1] comprises two stages, namely training and inference. In the training stage, the receiver first collects signals from the K end nodes operating in the IoT network. The signal collection procedure is elaborated in Section 3.2. The collected signals are stored as a training dataset, $\mathcal{D}^{train}$, given as

$$\mathcal{D}^{train} = \{(y_m, \mathbf{p}_m)\}_{m=1}^{M_{train}}, \tag{2}$$

where y_m is the m^{th} training sample and $\mathbf{p}_m$ is the corresponding one-hot encoded DUT label, given as

$$\mathbf{p}_m = O(\ell_m), \tag{3}$$

1. In some literature conventional RFFI refers to a system based on handcrafted features. In this paper, conventional RFFI refers to a typical deep learning-based system, in contrast to our proposed receiver-agnostic RFFI.

where $O(\cdot)$ denotes one-hot encoding operation, ℓ_m is the ground truth DUT label of the m^{th} training sample, M_{train} is the number of training samples. After building the training dataset, we define a neural network f and optimize its parameters Θ using $\mathcal{D}^{train}$ as defined below:

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \sum_{(y,\mathbf{p})\in\mathcal{D}^{train}} \mathcal{L}(f(y;\Theta), \mathbf{p}) \tag{4}$$

where $\mathcal{L}(\cdot)$ is the loss function that is usually cross-entropy in RFFI systems.

In the inference stage, the receiver captures a signal $y'(t)$ and feeds it into the well-trained neural network $f(\cdot;\Theta)$ for prediction. A probability vector $\hat{\mathbf{p}}$ is obtained via inference and is mathematically defined as

$$\hat{\mathbf{p}} = f(y';\Theta), \tag{5}$$

where $\hat{\mathbf{p}} = \{\hat{p}_1, ..., \hat{p}_k, \hat{p}_K\}$ is a probability vector over all the K DUTs, and $\hat{p}_k$ is the estimated probability for the k^{th} DUT. The predicted transmitter label, $\hat{\ell}$, is derived by simply selecting the index of the element with the highest probability as defined below:

$$\hat{\ell} = \underset{k}{\operatorname{argmax}}(\hat{\mathbf{p}}). \tag{6}$$

2.2 Problem Statement

As highlighted in the introduction, the receiver effect $\mathcal{G}'(\cdot)$ during inference is probably different from the receiver distortion $\mathcal{G}(\cdot)$ during training. In this case, the signal captured by the new receiver, $y'(t)$, has different characteristics from the training signals, $y(t)$. This distribution shift violates the basic independent and identically distributed (i.i.d) assumption of deep learning. For instance, when we feed $y'(t)$ collected by another receiver into the neural network to make a prediction, then the predicted label, $\hat{\ell}$, is given as:

$$\hat{\ell} = \underset{k'}{\operatorname{argmax}}(f(y';\Theta)) \tag{7}$$

which is probably different from the true label, ℓ. This can lead to misclassification.

As will be experimentally demonstrated in Section 5, both changing a new receiver for inference and the drift of receiver features over time can seriously degrade RFFI performance. A solution capable of training receiver-agnostic neural networks is urgently needed.

3 Receiver-Agnostic and Collaborative RFFI System

3.1 System Overview

This paper presents a receiver-agnostic and collaborative RFFI system. It involves two essential stages, as shown in Fig. 2(a) and 2(b), namely training a receiver-agnostic neural network and collaborative inference of multiple receivers. There is also an optional fine-tuning stage shown in Fig. 2(c). These stages are summarized below.

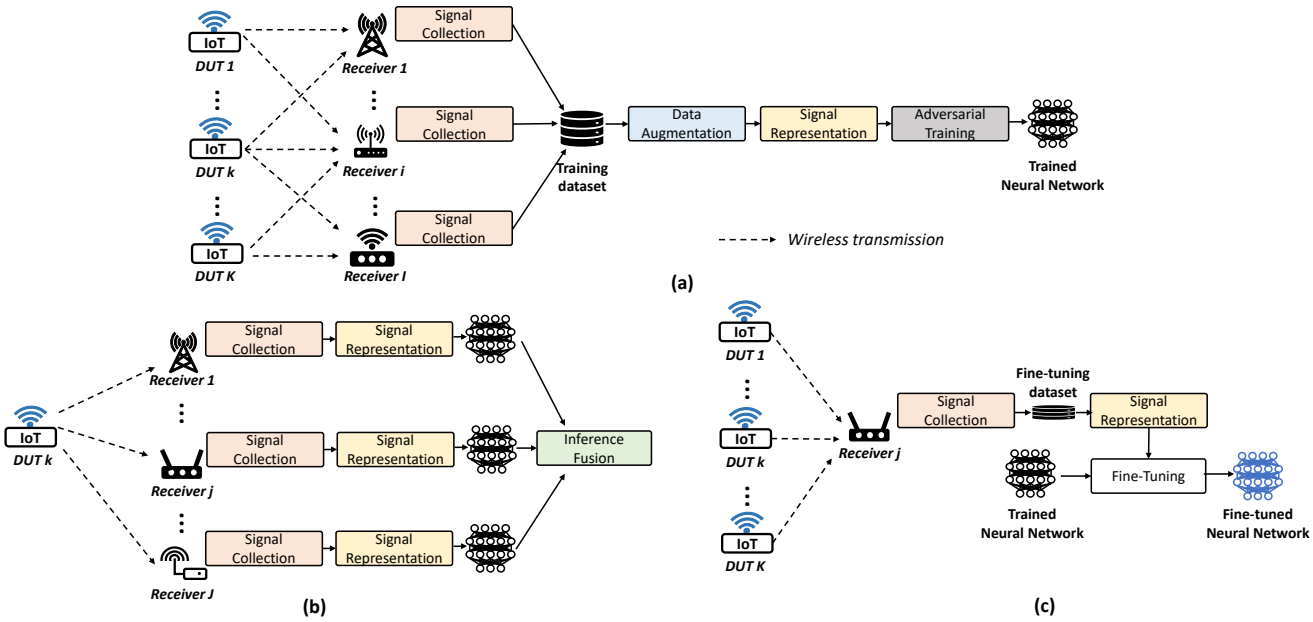


Fig. 2: Overview of the proposed receiver-agnostic and collaborative RFFI system. (a) Training of a receiver-agnostic neural network. (b) Collaborative inference using multiple receivers. (c) Fine-tuning of a trained neural network.

3.1.1 Train a Receiver-Agnostic Neural Network

As discussed in Section 2.2, neural networks trained with a conventional approach suffer from varying receiver characteristics. Therefore, we propose to leverage adversarial training to obtain a receiver-agnostic neural network.

During the training stage, there are K DUTs and I training receivers. Each receiver carries out the signal collection (Section 3.2) separately to capture wireless transmissions from the DUTs within range. The captured signals, i.e. IQ samples, along with the transmitter and receiver labels, comprise the training dataset. The dataset is then augmented with a wireless channel simulator to improve the channel robustness of the neural network (Section 3.3). The augmented time-domain IQ samples can be converted to appropriate signal representations to be used as the input to the neural network (Section 3.4). We can then obtain a receiver-agnostic neural network by adversarial training (Section 3.5).

3.1.2 Collaborative Inference of Multiple Receivers

During the inference, K end nodes and J receivers are involved. Note that the J inference receivers can be different from the I training ones. Take the k^{th} end node as an example, its transmission will be captured by all receivers in the range thanks to the broadcast nature of wireless transmissions. Each receiver will be equipped with a receiver-agnostic neural network. They will first carry out inferences independently, and then the results will be fused to obtain a more reliable prediction of the transmitter label. This part will be explained in Section 3.6.

3.1.3 Fine-Tuning

The performance of the receiver-agnostic neural network can be further improved by fine-tuning. A few packets can be collected by the underperforming receiver to slightly update the parameters of the trained neural network, which will be elaborated in Section 3.7. Note that fine-tuning is optional, which can be adopted by receivers with low-end RF front-ends when further improvements are required.

3.2 Signal Collection

The wireless transmissions will be first captured by the antenna and then downconverted to the baseband. The baseband signal will then undergo several steps including synchronization & preamble extraction, frequency offset compensation, and signal normalization.

3.2.1 Synchronization & Preamble Extraction

Synchronization is used to find the accurate start of the received packet. We then extract the preamble part for RFFI, in order to prevent the neural network from learning the identifiable information contained in the packet header or payload.

3.2.2 Frequency Offset Compensation

There are two reasons for performing frequency offset compensation:

- The frequency offset feature is easy to spoof by simply changing the transmitter carrier frequency, making the system vulnerable to attacks [14], [25], [33].
- The frequency offset is sensitive to temperature changes. A slight temperature change may make the system fail to work properly [9], [10], [34].

3.2.3 Signal Normalization

This is a standard operation in deep learning-based RFFI systems to prevent the neural network from classifying devices based on the received power. The normalization is achieved by dividing the signal by its root mean square value.

The processed signals along with the transmitter and receiver labels are stored in the training dataset, given as

$$\mathcal{X}^{train} = \{(y_m, \mathbf{p}_m, \mathbf{q}_m)\}_{m=1}^{M_{train}}, \tag{8}$$

where y_m is the m^{th} training signal, i.e. IQ samples, and M_{train} is the total number of captured transmissions. $\mathbf{p}_m = \{p_1, ..., p_k, ..., p_K\}$ and $\mathbf{q}_m = \{q_1, ..., q_i, ..., q_I\}$ are the corresponding one-hot encoded transmitter and receiver labels, respectively. Note that $\mathcal{X}^{train}$ for receiver-agnostic training additionally includes receiver labels, $\mathbf{q}$, compared to the training dataset $\mathcal{D}^{train}$ for conventional RFFI systems given in (2).

3.3 Data Augmentation

As revealed in (1), variations of channel impulse response $h(t)$ can affect the received signal $y(t)$. As discussed in Section 2.2, the characteristic change of $y(t)$ can seriously degrade the RFFI performance. It is therefore necessary to mitigate the channel effects on the RFFI system.

Data augmentation has been widely used to address the channel problem in RFFI [11], [13], [16], [22], [24], [33]. The training signals are replicated and passed through a channel simulator to emulate multipath and Doppler effects. With data augmentation, the training process can cover as many channel distributions as possible that may be present during inference, thus improving the system's robustness to channel variations.

3.4 Signal Representation

After data augmentation, the signals y can be converted to suitable signal representations S as neural network inputs. There have been many types of signal representations in previous studies, such as error signal [25], channel independent spectrogram [11], frequency spectrum [8], and differential constellation trace figure [17], to name but a few. Note that the unprocessed IQ samples y can also serve as neural network inputs after separating the I and Q components as two independent dimensions.

After signal representation, the data used for adversarial training becomes $\{(S_m, \mathbf{p}_m, \mathbf{q}_m)\}_{m=1}^{R \cdot M_{train}}$, where S_m is the signal representation converted from y_m. R is the replication factor which indicates how many times the training set is enlarged during data augmentation.

3.5 Adversarial Training

Adversarial training is an effective method to solve the problem of data distribution shift in deep learning [35]. Specific to RFFI, we leverage it to guide the neural network to learn receiver-independent features. As shown in Fig. 3, there are three components in adversarial training, namely the feature extractor, transmitter classifier, and receiver classifier. Their parameters are updated in an adversarial approach.

3.5.1 Feature Extractor

The feature extractor, $g(\cdot)$, converts the input signal representation, S, to a feature vector, given as

$$feature = g(S; \theta), \tag{9}$$

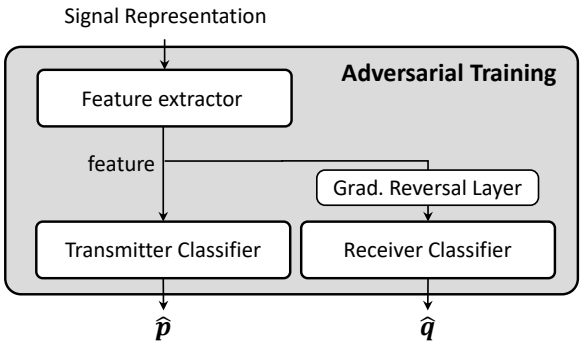


Fig. 3: Model architecture during adversarial training.

where θ denotes the learnable parameters[2] in the feature extractor which will be continuously updated during training. The feature extractor can be designed as a convolutional neural network (CNN), recurrent neural network (RNN) such as long short-term memory (LSTM) network, etc, depending on the characteristics of the input S. For example, CNN is usually good at processing images while RNN/LSTM performs well for time-series signals. The adversarial training aims to train a feature extractor whose output contains receiver-independent information.

3.5.2 Transmitter Classifier

The transmitter classifier accepts the extracted feature and makes predictions on the transmitter label. The output of the transmitter classifier, $\hat{\mathbf{p}}$, is a list of probabilities. The k^{th} element, $\hat{p}_k$, is mathematically given as

$$\hat{p}_k = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}}, \tag{10}$$

where z_k is the output of the k^{th} neuron before the softmax activation. We use cross-entropy for the transmitter classifier loss $\mathcal{L}_{tx}$, which is widely adopted in classification problems and is mathematically given as

$$\mathcal{L}_{tx} = -\sum_{k=1}^{K} p_k \log(\hat{p}_k), \tag{11}$$

where p_k is the corresponding ground truth in $\mathbf{p}$. The adversarial training process aims to maximize the performance of the transmitter classifier, which can be achieved by minimizing the loss $\mathcal{L}_{tx}$.

3.5.3 Receiver Classifier

The receiver classifier predicts the receiver label from the extracted feature vector. Its loss is defined as cross-entropy as well, given as

$$\mathcal{L}_{rx} = -\sum_{i=1}^{I} q_i \log(\hat{q}_i), \tag{12}$$

where $\hat{q}_i$ is the estimated probability of the packet being captured by the i^{th} receiver and q_i is the ground truth. The adversarial training aims to restrict the performance of the receiver classifier, i.e., to make the feature vector receiver-independent. This can be accomplished by interfering with the minimizing process of the receiver loss $\mathcal{L}_{rx}$.

2. θ is different from the Θ in (4). θ only denotes the parameters of the feature extractor while Θ represents that of the entire neural network.

3.5.4 Gradient Reversal Layer

The gradient reversal layer was first proposed in [35] to address the distribution shift problem. It does not affect forward propagation and only takes effect during backpropagation. We leverage it to guide the feature extractor to learn receiver-independent features.

As discussed in Section 2, the objective of an RFFI system is to predict from which DUT the signal is sent, which is identical to the goal of the transmitter classifier. Therefore, the performance of the transmitter classifier should be maximized during training, which can be achieved by minimizing the transmitter classifier loss, $\mathcal{L}_{tx}$. In contrast, the goal of the receiver classifier is to predict the receiver label from the extracted feature vector. However, this conflicts with the objective of the proposed receiver-agnostic RFFI protocol since we expect the feature vector to not contain any receiver-related information. This is equivalent to ensuring that the receiver classifier cannot predict the correct receiver label from the feature extractor. Therefore, the performance of the receiver classifier should be restricted during training, which can be achieved by limiting the minimization process of $\mathcal{L}_{rx}$.

The gradient reversal layer can be used to limit the performance of the receiver classifier. In a standard gradient descent process, the θ in the feature extractor is updated by

$$\theta \leftarrow \theta - \mu(\frac{\partial \mathcal{L}_{tx}}{\partial \theta} + \frac{\partial \mathcal{L}_{rx}}{\partial \theta}), \tag{13}$$

where μ is the learning rate. The role of the gradient reversal layer is to multiply $\frac{\partial \mathcal{L}_{rx}}{\partial \theta}$ by a negative factor $-\lambda$ during backpropagation, modifying the updating process to

$$\theta \leftarrow \theta - \mu(\frac{\partial \mathcal{L}_{tx}}{\partial \theta} - \lambda\frac{\partial \mathcal{L}_{rx}}{\partial \theta}). \tag{14}$$

In other words, the parameters θ of the feature extractor are updated in the opposite direction as instructed by the receiver classifier. Therefore, the receiver classifier cannot be improved as its feedback is not correctly followed. With this adversarial approach, we can train a feature extractor to extract transmitter-specific but receiver-independent information. Note that the parameters of the feature extractor, transmitter classifier, and receiver classifier are updated simultaneously.

Once the training is completed, the receiver classifier is removed since we do not need to predict the receiver label in an RFFI system. In the inference stage, the transmitter classifier is directly connected to the feature extractor to predict the transmitter identity.

3.6 Collaborative Inference

As illustrated in Fig. 2(b), each receiver uses the signal collection module explained in Section 3.2 to capture wireless transmissions. The captured signal is then converted to signal representation S and fed into the receiver-agnostic neural network. The output of the neural network at receiver j is a list of probabilities $\hat{\mathbf{p}}^j$. After this, the predicted probability vector $\hat{\mathbf{p}}^j$ and the estimated SNR of the received packet, γ^j, are gathered for collaborative identification. This can be performed on a cloud server or a central node.

The predictions from all the J receivers are then fused. Two fusion schemes are proposed, namely soft fusion and adaptive soft fusion. In the soft fusion scheme, the predictions $\hat{\mathbf{p}}^j$ are directly summed, which is given as

$$\hat{\mathbf{p}}^{fused} = \frac{1}{J}\sum_{j=1}^{J}\hat{\mathbf{p}}^j, \tag{15}$$

where $\hat{\mathbf{p}}^{\mathbf{fused}} = \{\hat{p}_1^{fused}, ..., \hat{p}_k^{fused}, ..., \hat{p}_K^{fused}\}$ is the fusion result. While in the adaptive soft fusion scheme, inferences from gateways with higher SNRs are assigned higher weights in the fusion process, which is mathematically expressed as

$$\hat{\mathbf{p}}^{fused} = \frac{1}{J}\sum_{j=1}^{J}\frac{\gamma^j}{\sum_{j=1}^{J}\gamma^j}\hat{\mathbf{p}}^j. \tag{16}$$

After fusion, the label corresponding to the highest value in $\hat{\mathbf{p}}^{\mathbf{fused}}$ is returned as the final prediction, formulated as

$$\hat{\ell}^{fused} = \underset{k}{\operatorname{argmax}}(\hat{\mathbf{p}}^{fused}), \tag{17}$$

where $\hat{\ell}^{fused}$ is the final predicted label.

3.7 Fine-Tuning

The trained receiver-agnostic neural network performs well on most receivers. However, sometimes its performance is still not satisfactory. This issue can be alleviated by fine-tuning, which refers to slightly adjusting the parameters of the neural network to make it better suited to a new task. The procedure of fine-tuning is shown in Fig. 2(c).

We need to first identify the underperforming receiver j as the fine-tuning target, which can be accomplished by comparing the decoded transmitter software address, e.g., MAC address, with the predicted device identity. If there are a significant number of packets where the software addresses do not match the predicted labels, then receiver j is deemed underperforming and requires fine-tuning.

After determining the underperforming receiver j, we use it to collect very few labeled signals y^j and store them in a fine-tuning dataset $\mathcal{X}^{tune}$, given as

$$\mathcal{X}^{tune} = \left\{(y_m^j, \mathbf{p}_m)\right\}_{m=1}^{M_{tune}}, \tag{18}$$

where M_{tune} is the number of fine-tuning packets. Note that compared to the training dataset $\mathcal{X}^{train}$ in (8), $\mathcal{X}^{tune}$ does not contain receiver labels because no adversarial training is used during fine-tuning. It is also worth noting that the labels $\mathbf{p}_m$ in $\mathcal{X}^{tune}$ are converted from the decoded transmitter software addresses, e.g., MAC addresses, and are not related to the predictions made by the underperforming neural network. These packets are too few to train a neural network from scratch, but sufficient for fine-tuning. Then we convert y^j to the selected signal representation S^j, and the data served for fine-tuning becomes $\left\{(S_m^j, \mathbf{p}_m)\right\}_{m=1}^{M_{tune}}$. With these data, the parameters of the receiver-agnostic neural network can be adjusted with a low learning rate. The cross-entropy loss given in (11) is used during fine-tuning. Once fine-tuning is completed, the fine-tuned neural network is updated at the underperforming receiver to obtain higher classification performance.

An important note is that fine-tuning needs to be carried out in a controlled environment, e.g., labs without attackers

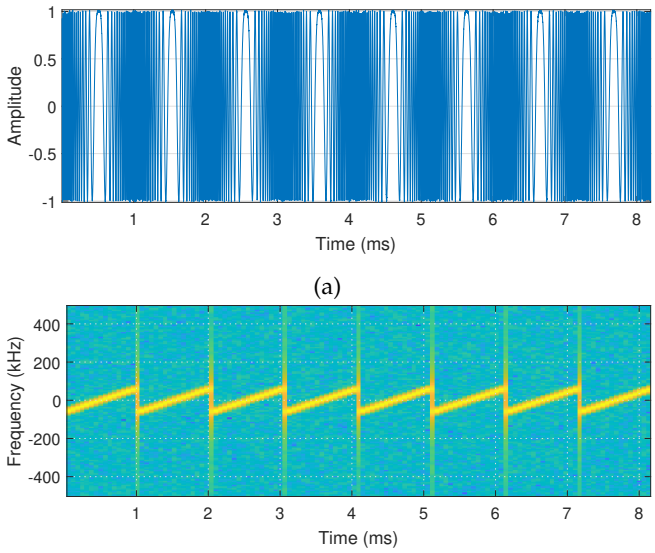


Fig. 4: The preamble part of a LoRa packet. (a) In-phase component. (b) Spectrogram generated by STFT.

present, so that the decoded MAC addresses can be considered reliable. Otherwise, the procedure of detecting underperforming receivers and fine-tuning may be interfered with by spoofers.

4 Case Study: LoRa/LoRaWAN-based Implementation

4.1 LoRa/LoRaWAN Primer

4.1.1 LoRa Physical Layer

LoRa is a low-power wide-area network (LPWAN) modulation technique based on chirp spread spectrum (CSS) technology. The information is encoded in a modulated linear chirp whose frequency changes linearly within a LoRa symbol. Due to the frequency-varying nature of LoRa signals, the spectrograms derived from time-frequency analysis algorithms such as short-time Fourier transform (STFT) are often used for representation/visualization [9], [36], [37]. The preamble part of a LoRa packet and its spectrogram is shown in Fig. 4, respectively. The preamble part of a LoRa packet contains eight unmodulated LoRa symbols.

4.1.2 LoRaWAN Star-of-Stars Topology

LoRaWAN defines a star-of-stars network topology, which is shown in Fig. 5. It allows one LoRa end node to establish wireless communication links (dashed lines) with multiple gateways at the same time. The LoRa gateways are connected to a server by using, e.g., WiFi, cellular communications, or Ethernet (solid lines). They relay the captured messages to the network server for collaborative decoding. The server runs network and application layer protocols.

The star-of-stars topology of LoRaWAN is suitable for implementing and exploring our proposed receiver-agnostic and collaborative RFFI protocol. Firstly, there are multiple gateways in a LoRaWAN network, and we desire a receiver-agnostic neural network that can be directly equipped on

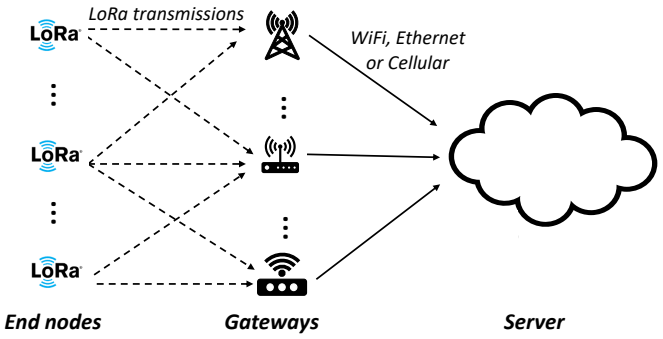


Fig. 5: LoRaWAN star-of-stars topology.

all LoRa gateways. Secondly, LoRaWAN already allows one LoRa transmission to be captured by multiple gateways. Applying collaborative RFFI to LoRa/LoRaWAN does not require any changes to the existing communication protocol. Moreover, the LoRa network server has vast computing resources thus the computing-expensive, receiver-agnostic training can be done efficiently.

4.2 Signal Collection

The LoRa signal collection is implemented using existing algorithms. The used LoRa packet synchronization algorithm is presented in [38], and LoRa frequency offset estimation and compensation algorithms are introduced in [9], [10].

4.3 Data Augmentation

The collected LoRa signals are replicated and passed through a channel simulator that emulates the multipath and Doppler effects in real communication scenarios. The used channel simulator is the same as that in [11], which uses the tapped delay line (TDL) model. More specifically, the exponential power delay profile (PDP) is employed to characterize the multipath effect, which is mathematically given as

$$E(p) = \frac{1}{\tau} e^{-pT_s/\tau} \tag{19}$$

where $E(p)$ denotes the power of the p^{th} path and T_s is the receiver sampling rate. τ is the root mean square delay spread, which is set randomly in the range of [5 ns, 300 ns]. The Doppler effect of each path is depicted by the well-known Jakes's spectrum [39], given as

$$J(f) = \frac{1}{\pi f_m \sqrt{1-(f/f_m)^2}}. \tag{20}$$

f_m denotes the maximum Doppler shift, which is uniformly distributed in the range of [0 Hz, 10 Hz]. The AWGN is also added to the training signals to improve the system performance in low SNR scenarios. The augmented signals with AWGN are uniformly distributed in the range of [0 dB, 50 dB].

4.4 Signal Representation

The channel-independent spectrogram proposed in [11] is used as the signal representation. The reason is twofold. Firstly, it is an effective approach to mitigate channel effects,

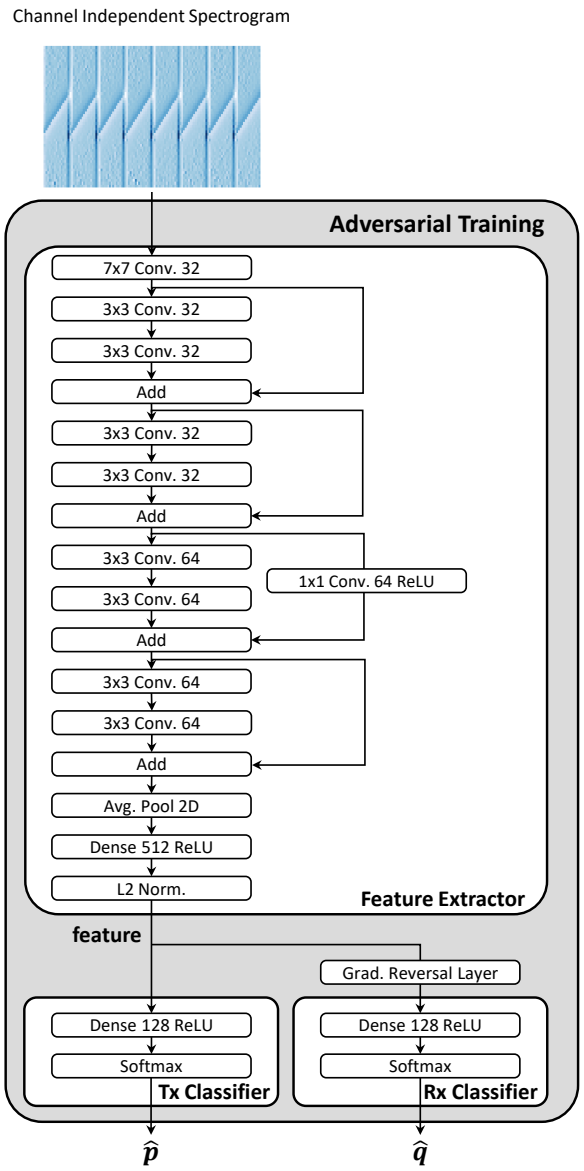


Fig. 6: Model architecture during adversarial training for the LoRa case study.

which has been experimentally verified in [11]. Secondly, it is a time-frequency representation that is suitable for CSS modulation [37]. Our prior work compares the signal representations in time, frequency, and time-frequency domains, concluding that time-frequency domain representation is the most appropriate for LoRa signals [9].

A channel-independent spectrogram of the preamble part of a LoRa packet is shown in Fig. 6 as the input to the neural network. In this paper, it is calculated with a window length of 128 and an overlap of 64. Interested readers should refer to [11] for more details.

4.5 Adversarial Training

The neural network shown in Fig. 6 is designed to process the converted channel independent spectrograms. The feature extractor refers to the neural network designed in [11]. It has nine convolutional layers with skip connections and each activated by a ReLU function. We perform L2 normalization on the extracted feature vector, which can increase system performance [28]. Both the transmitter and receiver classifiers consist of a 128-neuron dense layer activated by the ReLU function and a softmax layer for classification.

4.6 Collaborative Inference

The architecture of LoRaWAN is suitable for collaborative RFFI. When multiple LoRa gateways capture one packet, they perform independent inference using the receiver-agnostic neural network. The predictions and estimated SNRs are uploaded to the LoRaWAN network server and then fused to make a more reliable inference.

4.7 Fine-Tuning

Fine-tuning can be leveraged to improve the performance of underperforming LoRa gateways, as detailed in Section 3.7. Fine-tuning is feasible on edge LoRa gateways because the fine-tuning dataset is small and it can stop within a few training epochs. Nevertheless, we must recall that this is an optional process.

5 Evaluation of the Receiver-Agnostic and Collaborative Protocol

In this section, we experimentally evaluate the performance of the proposed receiver-agnostic and collaborative RFFI system in controlled environments, using the LoRa-based case study implementation.

5.1 Experimental Setup

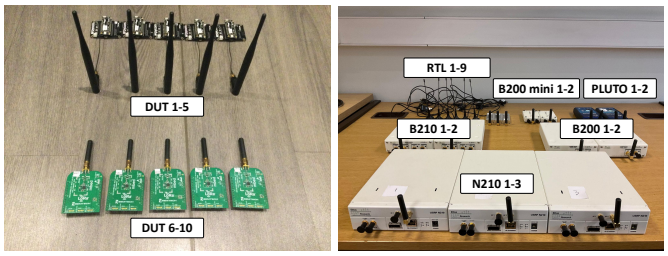
5.1.1 Device Information

We used ten LoRa devices as the DUTs to be identified and 20 SDR platforms to emulate the LoRa receivers/gateways.

- LoRa DUT: As shown in Fig. 7(a), we employed ten LoRa devices of two models, i.e., five of LoPy4[3] and five of mbed SX1261[4]. We configured all LoRa DUTs with a spreading factor of seven and bandwidth of 125 kHz. The carrier frequency was set to 868.1 MHz.
- SDR Receiver: As shown in Fig. 7(b), 20 SDR platforms of six models were used to investigate the receiver effect. The detailed SDR information is given in Table 1. These SDR platforms are made of different RF chipsets and analog-to-digital converters (ADCs) of different resolutions so that their hardware characteristics can be considered distinct. We used MATLAB Hardware Support for SDR[5] to access the SDR receivers. Though the MATLAB drivers for SDR receivers are different, all the SDRs run the same code to perform the signal collection procedure introduced in Section 3.2. Their receiver sampling rates were all set to 1 MHz.

Note that we collect packets from ten DUTs with each of the 20 SDR receivers. Therefore, there are 200 DUT-SDR pairs in total in the dataset.

3. https://pycom.io/product/lopy4/
4. https://os.mbed.com/components/SX126xMB2xAS/
5. https://www.mathworks.com/discovery/sdr.html



(a) (b)

Fig. 7: Experimental devices. (a) Ten LoRa DUTs. (b) 20 SDR receivers.

TABLE 1: Software-Defined Radios Receivers.

SDR name	Model	ADC	RF Chipset
RTL-1 ∼ RTL-9	RTL-SDR	8 bit	RTL2832U
PLUTO-1, PLUTO-2	ADALM-PLUTO	12 bit	AD9363
B200-1, B200-2	USRP B200	12 bit	AD9364
B200 mini-1, B200 mini-2	USRP B200 mini	12 bit	AD9364
B210-1, B210-2	USRP B210	12 bit	AD9361
N210-1 ∼ N210-3	USRP N210	14 bit	UBX RF Daughterboard

5.1.2 Dataset Description

We collected datasets from various SDRs to evaluate the receiver effect. In this section, all the datasets were collected in a typical residential room with line-of-sight (LOS) between the DUT and SDR receiver. The distance between the DUT (transmitter) and SDR (receiver) was one meter, and the received signals had a quite high SNR of above 50 dB. The transmitter and receiver locations are kept unchanged during all the collections. In such controlled environments, the wireless channel remained almost constant. This allowed investigation of the algorithm performance with minimal influence from channel effects. 800 packets were captured from each LoRa DUT-SDR pair, which were then pre-processed and augmented to construct the training dataset. Every test dataset contained 100 packets from each DUT-SDR pair. The detailed descriptions are given in the following subsections.

The controlled experiments in this section allow us to investigate the impact of the receiver without being significantly affected by the multipath effects. Additionally, the experimental setup enables us to precisely control the SNR of the test data, which is advantageous for evaluating the collaborative RFFI protocol.

5.1.3 CNN Training Configuration

We trained various CNN models with different configurations, which differ in two aspects, namely the number of training receivers I, and the types of training receivers. Since each subsection serves a specific evaluation purpose, we trained different CNN models with specially collected training datasets and evaluated them with corresponding test datasets. A summary is given in Table 2. It is worth noting that all the trained CNN models have the same structure, as introduced in Section 3.5.

The number of training receivers, I, is increased from one to five. When $I = 1$, the training can be simplified to the conventional approach introduced in Section 2. In this case, the deep learning model is constructed by directly connecting the feature extractor and transmitter classifier shown in Fig. 6.

When there are multiple receivers in the training dataset, we further divide the adversarial training into two categories, namely homogeneous and heterogeneous schemes, based on the diversity of the I training receivers.

- Homogeneous training: a low diversity of the I training receivers, i.e., the hardware characteristics of the I receivers are similar to each other.
- Heterogeneous training: a high diversity of the I training receivers, the hardware characteristics of the I receivers are significantly different from each other.

For example, when $I = 5$, homogeneous training receivers are all RTL-SDRs while heterogeneous training receivers are deliberately selected from different SDR models, namely RTL-1, PLUTO-1, B200-1, B200 mini-1, and B210-1.

5.1.4 Training Hyperparameters

All the CNN models in this paper were trained with the same settings. 10% of the training data was split out for validation. The CNN parameters were optimized by the stochastic gradient descent (SGD) optimizer (momentum 0.9) with an initial learning rate of 0.001 and a batch size of 64. The validation loss was monitored during training, and the learning rate was reduced by a factor of 0.2 when the validation loss did not drop within 10 epochs. Training stopped when the validation loss did not change within 20 epochs. The deep learning model was implemented using the TensorFlow library.

When fine-tuning was adopted, a lower learning rate of 0.00001 and a smaller batch size of 32 were used. The fine-tuning process stopped after 20 epochs and no learning rate scheduler was employed.

5.1.5 Evaluation Metric

The RFFI system is evaluated using the overall classification accuracy, which is calculated by dividing the number of correctly predicted packets by the total number of packets, given as

$$Accuracy = \frac{\text{Number of correctly classified packets}}{\text{Total number of packets}}. \quad (21)$$

5.2 Evaluation of Conventional Training

In this section, the impact of the receiver on RFFI systems is experimentally examined. It is found that the hardware characteristics of low-cost SDRs drift over time, making the conventional RFFI system unstable. Moreover, changing another receiver for signal acquisition also results in serious performance degradation. As only one receiver is involved during training and inference, all the CNN models used in this subsection are trained with the conventional scheme.

TABLE 2: Summary of Experimental Evaluation

Section	Purpose	Training Data	Test Data
5.2.1	The effect of receiver drift on RFFI	RTL-1 Day 1	RTL-1 Day 1, Day 2, Day 3, Day 4
		N210-1 Day 1	N210-1 Day 1, Day 2, Day 3, Day 4
5.2.2	The effect of receiver change on RFFI	RTL-1	Twenty SDRs
		N210-1	Twenty SDRs
5.3.1	The effect of receiver-agnostic training on receiver drift problem	RTL-1 to RTL-5	RTL-1 Day 1, Day 2, Day 3, Day 4
		RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	N210-1 Day 1, Day 2, Day 3, Day 4
5.3.2	The effect of receiver-agnostic training on receiver change problem	RTL-1 to RTL-5	Twenty SDRs
		RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	Twenty SDRs
5.3.3	Impact of number of training receivers	Different number of RTL-SDRs	N210-1
		Different number of SDRs of various models	N210-1
5.3.4	The effect of fine-tuning	RTL-1 to RTL-5	Twenty SDRs
		RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	Twenty SDRs
5.3.5	Performance on low-end receivers	RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	RTL-1 to RTL-9
		PLUTO-1, B200 -1, B200 mini-1, B210-1, N210-1	RTL-1 to RTL-9
5.4.1	Collaborative RFFI in a balanced SNR scenario	RTL-1 to RTL-5	Data from seven SDRs with various SNRs
		RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	Data from seven SDRs with various SNRs
5.4.2	Collaborative RFFI in an imbalanced SNR scenario	RTL-1 to RTL-5	Data from N210-1 to N210-3
		RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	Data from N210-1 to N210-3
6.2	The effect of collaborative RFFI in real environments	RTL-1, PLUTO-1, B200 -1, B200 mini-1, B210 -1	Data collected at six locations

5.2.1 *Receiver Drift*

We select RTL-6 and N210-1 to represent low-end and high-end SDR platforms, respectively, to study the receiver drift problem. We specifically train two CNNs and test them with the datasets collected on four continuous days:

- Training dataset from RTL-6 Day 1. Test datasets from RTL-6 Day 1, Day 2, Day 3, and Day 4.
- Training dataset from N210-1 Day 1. Test datasets from N210-1 Day 1, Day 2, Day 3, and Day 4.

The classification results are given in Fig. 8(a). It can be observed that the CNN trained with N210-1 is relatively stable over time, with its accuracy remaining nearly unchanged during the four days. There are likely two reasons for this result. Firstly, the transmitter and receiver locations are kept unchanged during all the collections, thus the channel variation is not severe. Secondly, the stability should also be attributed to the use of CFO compensation, data augmentation, and channel-independent spectrogram. More specifically, the CFO compensation prevents the performance degradation caused by the frequency change of transmitter oscillators [9]. While the data augmentation and channel-independent spectrogram prevent the degradation caused by the wireless channel changes [11].

In contrast, the performance of RTL-6 degrades seriously by around 40% on Day 2, 3, and 4. The experimental settings are exactly the same except for the receiver, therefore we reckon the performance difference is due to the unstable hardware characteristics of RTL-SDR. The features of RTL-6 on Day 2, 3, and 4 may be different from Day 1.

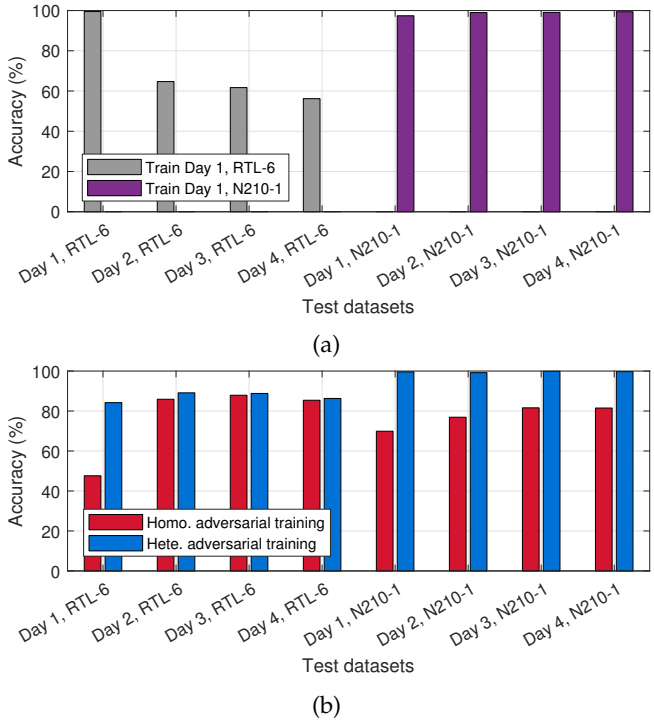


Fig. 8: Evaluation of the receiver drift problem. (a) Conventional training. (b) Receiver-agnostic training.

5.2.2 *Receiver Change*

As discussed in Section 2, using different receivers for training and inference can lead to a sharp performance decline. To study the receiver change problem, we made the following evaluation:

- Training dataset from RTL-1. Test datasets from the 20 different SDR receivers.
- Training dataset from N210-1. Test datasets from the

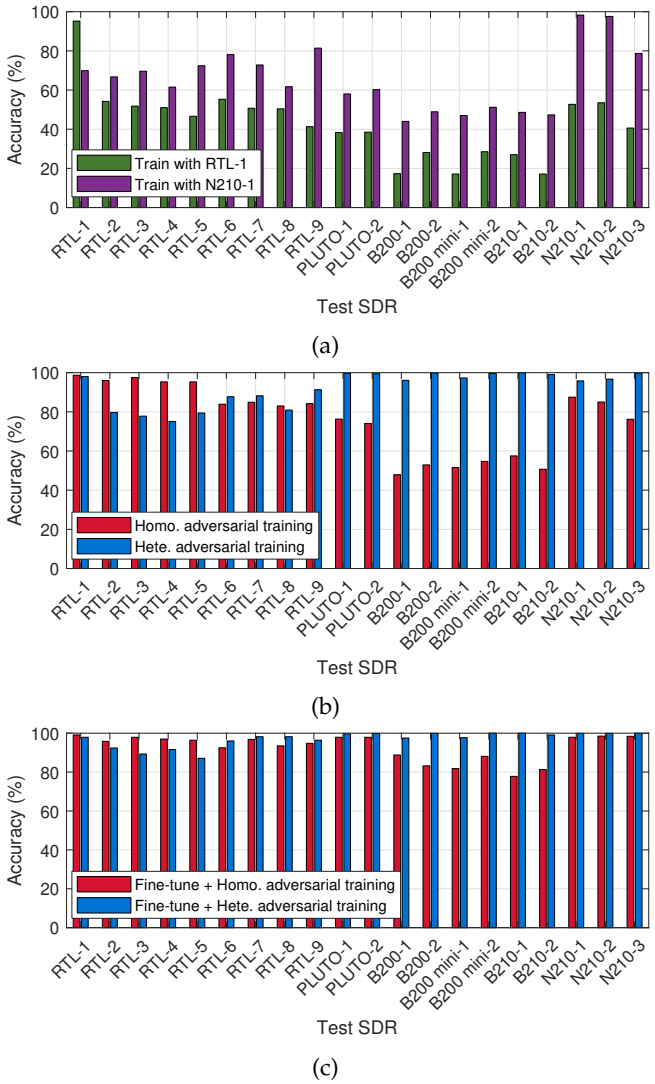


Fig. 9: Evaluation of the receiver change problem, the effect of receiver-agnostic training, and fine-tuning. Test on 20 different receivers. (a) Conventional training. (b) Receiver-agnostic training. (c) Receiver-agnostic training with fine-tuning.

20 different SDR receivers.

As illustrated in Fig. 9(a), the accuracy is high only when the same receiver is used for training and testing. The CNN trained with RTL-1 performs poorly on other receivers, especially on USRP B-series, i.e., B200, B200 mini, and B210. The B200-1 leads to the worst result, with only 20% accuracy. It drops 70% compared to the result with RTL-1. In comparison, the CNN trained with N210-1 performs slightly better. Its performance does not degrade on N210-2. This is likely because the receiver distortion of USRP N210 devices is not severe as relatively expensive hardware components are used. Therefore, N210-2 probably has similar hardware characteristics to the training receiver N210-1. Beyond that, we can still observe a significant accuracy decline in other SDR receivers, including the N210-3 from the same model. This highlights the significance of designing receiver-agnostic RFFI protocols.

5.3 Evaluation of Receiver-Agnostic Training

As discussed in Section 5.2, the drift of hardware characteristics of low-cost receivers can compromise RFFI stability. In addition, changing another receiver for RFFI also reduces system performance. The receiver-agnostic training can mitigate performance reduction effectively. We train two CNN models using homogeneous and heterogeneous adversarial training strategies, respectively. Five training receivers are involved unless otherwise stated. The training and test configuration is emphasized in each individual subsection.

5.3.1 Effect on Receiver Drift

First, we evaluate the performance of receiver-agnostic training on the receiver drift problem. The following evaluation schemes were designed:

- Train with the homogeneous scheme (RTL-1 to RTL-5). Test on RTL-6 and N210-1 for four consecutive days.
- Train with the heterogeneous scheme (RTL-1, PLUTO-1, B200-1, B200 mini-1, B210-1). Test on RTL-6 and N210-1 for four consecutive days.

The results are given in Fig. 8(b). It can be observed that the CNN trained with the heterogeneous scheme (blue bars) performs well on all the test datasets. Moreover, we do not observe any significant performance variation on the RTL-6 datasets collected over four continuous days, indicating that the system is relatively stable after employing the receiver-agnostic training. In contrast, the CNN trained with a homogeneous scheme is still unsatisfactory. In particular, its performance drifts severely on RTL-6 datasets, from 47.6% on Day 1 to 87.9% on Day 3. This is likely because the hardware characteristic of RTL-6 is unstable, which has been experimentally demonstrated and discussed in Section 5.2.1. More specifically, the homogeneously trained CNN has poor generalization ability and cannot generalize well to Day 1 RTL-6 characteristics. Therefore, the heterogeneous scheme is more recommendable, as the diversity of training receivers leads to better generalization ability.

5.3.2 Effect on Receiver Change

The proposed receiver-agnostic training can mitigate the performance degradation caused by receiver changes. In other words, the CNN trained with receiver-agnostic training can be directly applied to new receivers that are not included in the training process. The following evaluations were conducted:

- Train with the homogeneous scheme (RTL-1 to RTL-5). Test on the 20 different SDR receivers.
- Train with the heterogeneous scheme (RTL-1, PLUTO-1, B200-1, B200 mini-1, and B210-1). Test on the 20 different SDR receivers.

The results of receiver-agnostic training are presented in Fig. 9(b). The accuracy of all test datasets is greatly improved compared to the results of conventionally trained CNNs in Fig. 9(a). The accuracy is always above 75% for the CNN trained with the heterogeneous scheme. We can also find that the heterogeneous training scheme performs better than the homogeneous one, except on RTL-2 to RTL-5. The

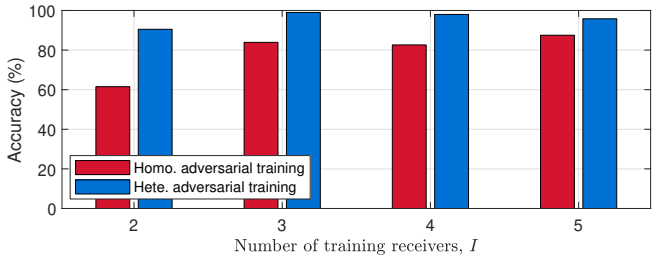


Fig. 10: The effect of the number of training receivers I on RFFI performance. Test on N210-1, not included in the I training receivers. Both homogeneous and heterogeneous strategies are evaluated.

reason for the exception is that RTL-2 to RTL-5 are included during homogeneous training but not in heterogeneous training, as illustrated in Table 2. The CNN trained with a homogeneous scheme does not generalize well to USRP B series SDRs, i.e., B200, B200 mini, and B210, which is likely because the hardware difference is huge among the training RTL-SDRs and the testing USRP B-series SDRs.

5.3.3 Impact of the Number of Training Receivers

We train CNNs with different numbers of receivers both for homogeneous and heterogeneous schemes. These CNNs are then tested on N210-1 that is not included in the I training receivers. As the result given in Fig. 10, in both homogeneous and heterogeneous training, the classification accuracy benefits from the increase in the number of training receivers I. However, the improvement is marginal after I reaches three. The heterogeneous scheme finally achieves higher accuracy than the homogeneous one. Note that in the heterogeneous setting, the slight performance degradation when the number of training receivers increases from three to five is likely caused by the training randomness introduced during parameter initialization, optimization, etc.

5.3.4 Effect of Fine-Tuning

As revealed in Fig. 9(b), although the CNNs trained with a receiver-agnostic scheme can be directly deployed on a new receiver, they still cannot achieve satisfactory performance in some cases. Fine-tuning the trained CNN model can further improve the performance of new receivers. We use 20 packets from each DUT-SDR pair for fine-tuning and the results are shown in Fig. 9(c). It is clear that fine-tuning leads to significant improvements of up to 40% compared to the results in Fig. 9(b). It was also found that the homogeneous scheme still underperforms the heterogeneous counterpart even after fine-tuning.

We further investigate the effect of the number of packets used for fine-tuning. Three fine-tuning cases are evaluated:

- Fine-tune the CNN trained with a homogeneous scheme. Training receivers are RTL-1 to RTL-5.
- Fine-tune the CNN trained with a heterogeneous scheme. Training receivers are RTL-1, PLUTO-1, B200-1, B200 mini-1, B210-1.
- Fine-tune the CNN trained with a conventional scheme. The training receiver is B200-1.

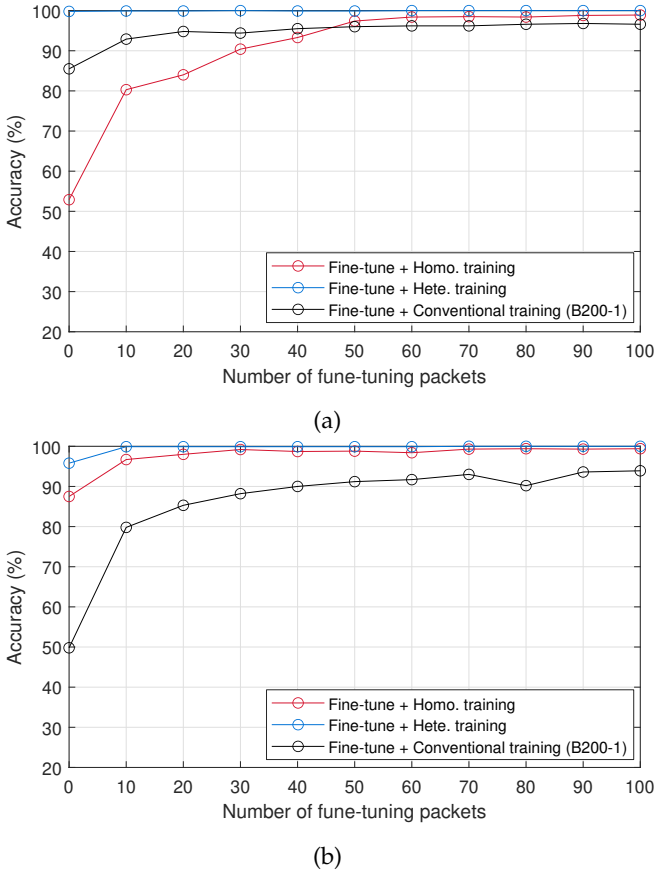


Fig. 11: The effect of the number of fine-tuning packets. Three training schemes are evaluated, namely homogeneous, heterogeneous, and conventional training. Fine-tuning is not employed when the number of packets is zero. (a) Test on B200-2. (b) Test on N210-1.

The test receivers for these cases are B200-2 and N210-1. More specifically, we collect different amounts of packets with B200-2 and N210-1 to fine-tune the CNNs and then evaluate the performance after fine-tuning. Note that both B200-2 and N210-1 are not included during the training process of any cases, thus we are evaluating the RFFI performance on new receivers.

The effect of the number of fine-tuning packets is illustrated in Fig. 11. It can be seen that fine-tuning the heterogeneously trained CNN requires the least number of packets, i.e., less than 10, and can achieve the best performance. For the B200-2 receiver, fine-tuning on the basis of the conventional scheme requires fewer packets than the homogeneous scheme. The reason is the conventional scheme uses the B200-1 for training, which has similar hardware characteristics to the test receiver B200-2. This simplifies the fine-tuning process. For the N210-1 receiver, fine-tuning on the basis of both heterogeneous and homogeneous schemes can reach above 95% accuracy with 10 packets, but a similar accuracy cannot be achieved with a conventionally trained CNN. In addition, more packets are required when fine-tuning is performed on the basis of a conventional training scheme. Taking into account the above analysis and discussion, it can be concluded that fine-tuning on the basis of the heterogeneous scheme is the optimal solution in terms of

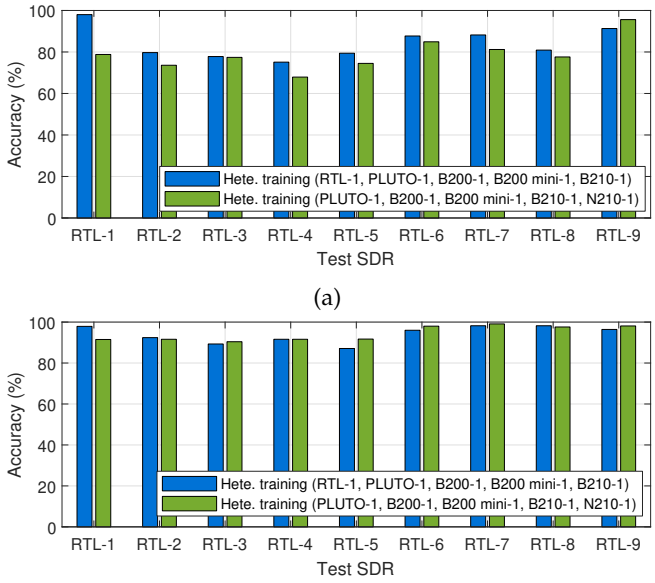


Fig. 12: The performance of receiver-agnostic NN on low-end receivers, i.e., RTL-1 to RTL-9. Two heterogeneous training configurations are investigated but with different training receivers. The effect of fine-tuning is also investigated. (a) Heterogeneous training. (b) Heterogeneous training with fine-tuning.

performance and the number of required packets.

5.3.5 Performance on Low-End Receivers

This section focuses on investigating the performance of the receiver-agnostic neural network on low-end receivers, i.e., RTL-SDRs. Two heterogeneous training configurations are considered:

- Training receivers include RTL-1, PLUTO-1, B200-1, B200 mini-1, B210-1. Test on RTL-1 to RTL-9.
- Training receivers include PLUTO-1, B200-1, B200 mini-1, B210-1, N210-1. Test on RTL-1 to RTL-9.

The difference between the two configurations is whether the low-end RTL-SDRs are used during training. In the second configuration, the RTL-series used for testing is not present during training and therefore requires a stronger generalization capability of the receiver-agnostic neural network.

Fig. 12(a) illustrates the performance of the receiver-agnostic neural network on RTL-1 to RTL-9. It can be observed that the first training configuration (blue bars) performs slightly better than the second one (green bars). The reason is RTL-1 is included in the first training configuration thus the trained neural network can better generalize to the rest RTL-SDRs. For both configurations, an average accuracy of approximately 80% can be achieved, and Fig. 12(b) shows fine-tuning technique can further improve the performance by almost 10%. Therefore, it can be concluded that the RFFI system performs well on low-end receivers using the proposed receiver-agnostic protocol.

5.3.6 Summary

In conclusion, receiver-agnostic training can effectively mitigate the performance degradation caused by receiver drift and change. It is recommended to use a heterogeneous scheme since better generalization ability can be achieved. The RFFI performance degrades when a low-end RF front-end is used for signal capturing, e.g., RTL-SDRs. The results show that RFFI performance can be significantly improved with less than 50 packets from each DUT-SDR pair. Commercial IoT gateways are often equipped with a microcontroller for decoding and management. And many deep learning frameworks support on-device training on microcontrollers, e.g., TensorFlow Lite. In addition, the IoT gateways are often powered by mains but not batteries and have a sufficient energy supply for fine-tuning. Therefore, we believe the proposed fine-tuning technique can be applied to gateways with low-end RF front-ends to improve identification performance.

5.4 Evaluation of Collaborative RFFI

In this subsection, the proposed collaborative RFFI scheme is evaluated. Artificial AWGN is added to the test data to emulate signals collected at various SNRs.

5.4.1 Balanced SNR Scenario

We first consider a simple case where the signals collected by different receivers have the same SNR. In this case, the adaptive fusion in (16) can be simplified as the simple fusion in (15) since all receivers are assigned the same weights. Seven SDRs that are not included during the receiver-agnostic training are selected for evaluation, namely PLUTO-2, B200-2, B200 mini-2, B210-2, N210-1, N210-2, and N210-3. Then the CNN trained with the heterogeneous scheme is directly applied without fine-tuning. The classification results are shown in Fig. 13. It can be observed that the improvement becomes more significant as more receivers get involved in collaborative inference. When SNR is between 15-20 dB, the collaborative inference using seven SDR receivers performs 20% better than using an individual receiver.

5.4.2 Imbalanced SNR Scenario

A more common scenario in practice is that the SNRs of packets collected by different LoRa receivers are different. In this case, soft fusion and adaptive soft fusion may lead to different results.

We employ N210-1 to N210-3 for evaluation. The SNR of N210-3 was adjusted from 0 dB to 40 dB, while SNRs of N210-1 and N210-2 are fixed to 10 dB and 20 dB, respectively. The results are shown in Fig. 14. The accuracy of N210-3 gradually increases with SNR. It reaches close to 100% when the SNR of N210-3 is over 35 dB. We also show the average accuracy of N210-1 and N210-2 in the figure, which is around 40% and 60%, respectively.

In this imbalanced SNR scenario, both soft fusion and adaptive soft fusion schemes are effective when the SNR of N210-3 is below 25 dB. Their accuracy is always higher than any individual receiver. However, we can also see that adaptive soft fusion is less effective when the SNR of N210-3 is over 25 dB. The reason is the SNR of N210-3 is high and

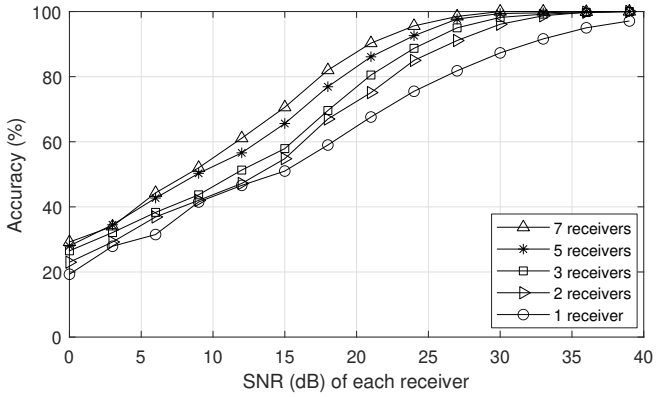


Fig. 13: Collaborative RFFI in a balanced SNR scenario. In this case, soft fusion is equivalent to adaptive soft fusion. The CNN is trained with a heterogeneous strategy without fine-tuning. Test on seven SDR receivers that are not included during training.

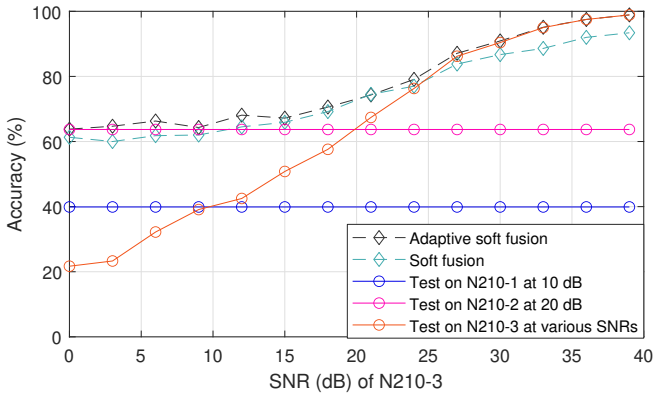


Fig. 14: Collaborative RFFI in an imbalanced SNR scenario. Three SDR receivers not included during training are used. The SNR of N210-3 is adjusted from 0 dB to 40 dB, while N210-1 and N210-2 are fixed at 10 dB and 20 dB, respectively.

the inferences from N210-1 and N210-2 are assigned very low weights. Compared to the adaptive soft fusion, when the SNR of N210-3 is over 25 dB, the soft fusion scheme without weighting leads to even lower accuracy than N210-3 itself. This indicates assigning weights according to SNR is necessary for the collaborative RFFI, thus the adaptive soft fusion scheme is recommended.

6 A Hybrid RFFI Protocol and Evaluation in a Real Scenario

In this section, a more practical RFFI testbed is established for evaluation. More specifically, three SDRs were positioned within an office building, collecting LoRa packets concurrently and executing collaborative inference to identify LoRa devices. To further enhance the system's performance, we combine the designed receiver-agnostic and collaborative scheme with other techniques, i.e., online augmentation and multi-packet inference proposed in our previous work [31] to construct a hybrid RFFI protocol.

6.1 A Hybrid RFFI Protocol

A hybrid RFFI protocol, which is resilient to noise contamination and receiver changes, has been designed for challenging RFFI working conditions. The receiver-agnostic RFFI scheme discussed in Section 3 and Section 4 focuses on solving the receiver problem and is not optimized for low-SNR scenarios. The RFFI system identifies devices by analyzing distortions caused by hardware imperfections. However, these distortions are subtle and difficult to detect in the received signal. This task is particularly challenging when the SNR is low, as the noise largely corrupts the distortion. In Section 5.4, collaborative inference has been proven to be an effective approach for mitigating noise corruption. To further enhance the system's ability to withstand noise interference, we additionally integrated online augmentation and multi-packet inference techniques proposed in [31] to create a hybrid RFFI protocol.

6.1.1 Online Augmentation

The online augmentation is an upgrade to the data augmentation module described in Section 3.3 and Section 4.3. In a nutshell, it is implemented by adding various levels of artificial noise to the mini-batches during training. The online approach is more effective than directly injecting noise into the dataset as described in Section 3.3, i.e., an offline manner, since it allows the neural network to learn a wider range of noisy signals during the training procedure.

6.1.2 Multi-Packet Inference

Multi-packet inference is another approach to withstand noise contamination as proposed in [31]. It is achieved by averaging the predictions for multiple wireless packets. The prediction $\hat{\mathbf{p}}^j$ at receiver j is mathematically given as

$$\hat{\mathbf{p}}^j = \frac{1}{D}\sum_{d=1}^{D}\hat{\mathbf{p}}_d^j \tag{22}$$

where $\hat{\mathbf{p}}_d^j$ is the prediction for the d^{th} packet with the same decoded transmitter address. D is the total number of packets involved in the multi-packet inference. This technique is lightweight and can be seamlessly combined with the receiver-agnostic and collaborative RFFI protocol.

6.2 Experimental Setup

6.2.1 Data Collection Settings

The floor plan of the experimental environment is given in Fig. 15. Three SDR receivers are used, i.e., N210-1, N210-2, and N210-3. N210-1 is placed in an office while N210-2 and N210-3 are placed in another meeting room. The LoRa DUTs are in turn located at six locations A-F. More specifically, we run the three SDR receivers concurrently to collect 300 packets from each DUT-SDR pair at one location and then repeat the collection after moving the DUTs to the next location. The average estimated SNRs of the collected packets at Locations A-F are shown in Fig. 16. It demonstrates that the estimated SNRs at Locations E and F are as low as 10 dB, implying that the hybrid RFFI protocol is needed to withstand noise contamination.

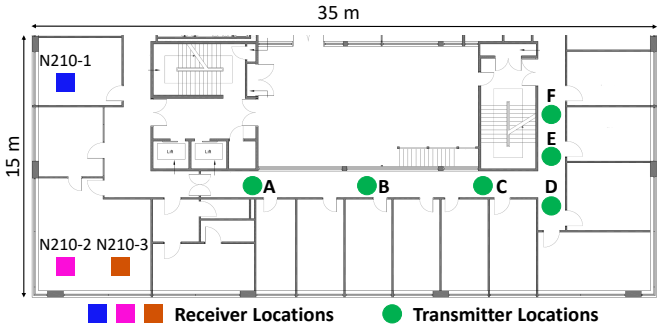


Fig. 15: Floor plan.

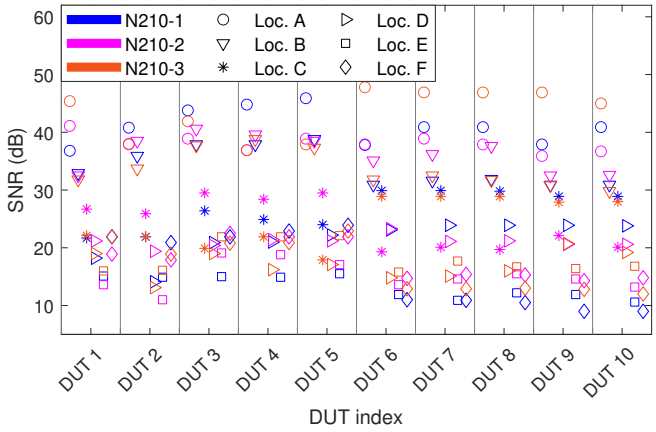


Fig. 16: Estimated SNRs of the received LoRa packets at locations A-F. Marker colors and symbols represent the SDR receiver and location, respectively.

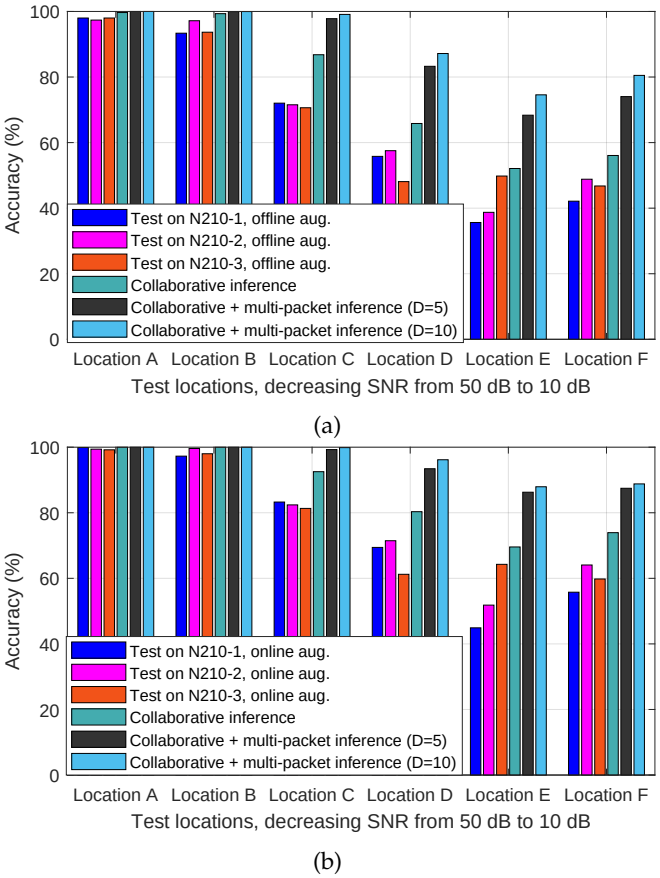


Fig. 17: The performance of the hybrid RFFI system. The DUTs are in turn placed at six locations. The SNR values at locations E and F are down to 10 dB. The CNN trained with a heterogeneous scheme is used without fine-tuning. (a) Offline augmentation is used. (b) Online augmentation is used.

6.2.2 *Training Configurations*

We use the CNN trained with the heterogeneous scheme for evaluation. The training dataset and hyperparameters are exactly the same as those described in Section 5.1.4. The only difference is that online augmentation is utilized during training to increase noise robustness. It is worth noting that the training data is collected in a residential room in an LOS scenario, which is different from the test environments.

6.3 Experimental Results

The performance of the hybrid RFFI protocol is demonstrated in Fig. 17. In terms of general trends, the identification accuracy gradually decreases as the transmitter is placed further away from the SDRs, i.e., Locations A to F. As shown in Fig. 17(a), the accuracy of an individual receiver, e.g., N210-1, drops from 100% (Location A) to around 40% (Location F). The main reason is that the average SNR descends from 50 dB at Location A to 10 dB at Location F, as illustrated in Fig. 16. Therefore, it is necessary to employ the hybrid RFFI protocol introduced in Section 6.1 to enhance identification performance.

The hybrid RFFI protocol leverages three techniques to withstand noise contamination, namely online augmentation, collaborative inference, and multi-packet inference. The results in Fig. 17(b) are achieved by utilizing online augmentation, which has roughly 10% improvement for an individual receiver as shown in Fig. 17(a). This demonstrates that online augmentation can effectively improve the noise robustness of the neural network. Figs. 17(a) and 17(b) show that the proposed collaborative inference scheme can further improve the identification accuracy by around 10%. For example, as shown in Fig 17(b), the accuracy is increased from around 60% to 70% at Location F when collaborative inference is applied. Additionally, the performance can be enhanced by up to 20% through employing the multi-packet inference method outlined in Section 6.1.2. The results indicate that involving only five packets during inference, i.e., D=5, can lead to a significant improvement.

In summary, the hybrid RFFI protocol extensively increases the identification accuracy, particularly in the low-SNR scenarios. As shown in Fig. 17(a), an individual receiver only achieves around 50% accuracy when the transmitters are positioned at location F. After leveraging the techniques of online augmentation, collaborative, and multi-packet inference, the accuracy is increased to approximately 90%, as depicted in Fig. 17(b). This indicates that a total performance improvement of 40% can be achieved through the implementation of the hybrid RFFI protocol.

7 Related Works

In recent years, RFFI has benefited greatly from the fast development of deep learning techniques. Most deep learning-based RFFI studies are devoted to leveraging advanced neural network models to better extract transmitter impairments. These models include CNN [5]–[17], [19], [22]–[25], [27], [28], [40], LSTM [9], [12], [22], [26], and multiple layer perceptron (MLP) [8], [9], [12], gated recurrent unit (GRU) models [12], etc. These models are effective in improving identification performance, however, there are still challenges that are overlooked in previous studies.

One challenge is that RFFI systems are affected by the receiver hardware characteristics as the captured physical layer signal is distorted by the receiver chain. To the best of the authors' knowledge, there have been few studies investigating the receiver effects. Zhang *et al.* [5] revealed how the change of receiver characteristics affects the RFFI performance, but the work is mostly simulation-based and does not present a countermeasure. Merchant *et al.* [41] undertook experiments using high-end receivers and observed the performance degradation caused by the receiver effect. However, the low-end receivers are not investigated. Elmaghbub *et al.* [27] experimentally revealed that using different receivers during training and inference degrades system performance, but no solutions are designed.

Leveraging multiple receivers in RFFI can enhance system performance. To the authors' best knowledge, there are only two papers that have investigated the RFFI using multiple receivers/antennas [18], [32]. Andrews *et al.* [32] have investigated how to combine the observations from multiple antennas and compared three combination methods, but it is based on traditional frequency features and is not available in deep learning-based RFFI systems. He *et al.* [18] employed a support vector machine (SVM), MLP, and LSTM to fuse the extracted decomposed features and compared the fusion performance. However, it is mainly based on simulation with limited experimental results. A collaborative RFFI scheme for deep learning-based approaches needs to be designed and experimentally evaluated.

8 Conclusions

In this paper, a receiver-agnostic and collaborative RFFI scheme is proposed, and LoRa/LoRaWAN is used as a case study for experimental evaluation. Experiments were conducted with ten COTS LoRa DUTs and 20 SDR receivers in both residential and office building environments. The results show that conventional deep learning-based RFFI systems are seriously affected by the changes in receiver hardware characteristics. We experimentally find that the performance of an RFFI system implemented with low-cost SDR receivers (RTL-SDR) drops by up to 40% over four continuous days. This may be due to the unstable characteristics of the hardware components in the RTL-SDR. We also find that changing a new SDR for signal collection results in a sharp decline in identification accuracy, up to 70% in some cases. To make the neural network receiver-agnostic, we leverage an adversarial approach during its training process. More specifically, a gradient reversal layer is employed to guide the neural network to learn receiver-independent features. We evaluate the receiver-agnostic neural network with 20 different SDR receivers and the identification performance is always maintained above 75%. Fine-tuning can be done by slightly adjusting the parameters of the neural network using a few collected packets, which can further improve the performance of the receiver-agnostic neural network. The experimental results show that fine-tuning can lead to up to 40% accuracy improvement. Collaborative RFFI with multiple receivers can enhance identification performance. The predictions made by individual receivers can be fused by weighted averaging. The results show that the collaborative RFFI can increase the identification accuracy by up to 20%. Finally, a more realistic experiment is conducted by deploying three USRP N210 SDRs in an office building. The receiver-agnostic neural network performs well on these SDRs and the collaborative inference can improve the identification accuracy by 10%. Future work should focus on improving identification performance in low-SNR, long-distance, and strong-multipath scenarios.

References

[1] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 2015.

[2] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[3] J. Zhang, G. Li, A. Marshall, A. Hu, and L. Hanzo, "A new frontier for IoT security emerging from three decades of key generation relying on wireless channels," *IEEE Access*, vol. 8, pp. 138 406–138 446, 2020.

[4] W. Wang, Z. Sun, S. Piao, B. Zhu, and K. Ren, "Wireless physical-layer identification: Modeling and validation," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 2091–2106, 2016.

[5] J. Zhang, R. Woods, M. Sandell, M. Valkama, A. Marshall, and J. Cavallaro, "Radio frequency fingerprint identification for narrowband systems, modelling and classification," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3974–3987, 2021.

[6] S. Hanna, S. Karunaratne, and D. Cabric, "Wisig: A large-scale WiFi signal dataset for receiver and channel agnostic RF fingerprinting," *IEEE Access*, vol. 10, pp. 22 808–22 818, 2022.

[7] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, K. Chowdhury, S. Ioannidis, and T. Melodia, "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 646–655.

[8] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, and B. Preneel, "Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning," in *Proc. ACM Conf. Security Privacy Wireless Mobile Netw. (WiSec)*, 2017, pp. 58–63.

[9] G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, "Radio frequency fingerprint identification for LoRa using deep learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2604–2616, 2021.

[10] ——, "Radio frequency fingerprint identification for LoRa using spectrogram and CNN," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Virtual Conference, May 2021, pp. 1–10.

[11] G. Shen, J. Zhang, A. Marshall, and J. R. Cavallaro, "Towards scalable and channel-robust radio frequency fingerprint identification for LoRa," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 774–787, 2022.

[12] D. Roy, T. Mukherjee, M. Chatterjee, E. Blasch, and E. Pasiliao, "RFAL: Adversarial learning for RF transmitter identification and classification," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 783–801, 2019.

[13] M. Cekic, S. Gopalakrishnan, and U. Madhow, "Wireless fingerprinting via deep learning: The impact of confounding factors," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, 2021, pp. 677–684.

[14] J. Yu, A. Hu, G. Li, and L. Peng, "A robust RF fingerprinting approach using multisampling convolutional neural network," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6786–6799, 2019.

[15] T. Jian, Y. Gong, Z. Zhan, R. Shi, N. Soltani, Z. Wang, J. G. Dy, K. R. Chowdhury, Y. Wang, and S. Ioannidis, "Radio frequency fingerprinting on the edge," *IEEE Trans. Mobile Comput.*, 2021.

[16] N. Soltani, G. Reus-Muns, B. Salehihikouei, J. Dy, S. Ioannidis, and K. Chowdhury, "RF fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15 518–15 531, 2020.

[17] L. Peng, J. Zhang, M. Liu, and A. Hu, "Deep learning based RF fingerprint identification using differential constellation trace figure," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1091–1095, 2019.

[18] B. He and F. Wang, "Cooperative specific emitter identification via multiple distorted receivers," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3791–3806, 2020.

[19] Y. Qian, J. Qi, X. Kuai, G. Han, H. Sun, and S. Hong, "Specific emitter identification based on multi-level sparse representation in automatic identification system," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2872–2884, 2021.

[20] J. Gong, X. Xu, and Y. Lei, "Unsupervised specific emitter identification method using radio-frequency fingerprint embedded InfoGAN," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2898–2913, 2020.

[21] S. Rajendran, Z. Sun, F. Lin, and K. Ren, "Injecting reliable radio frequency fingerprints using metasurface for the Internet of Things," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1896–1911, 2020.

[22] A. Al-Shawabka, P. Pietraski, S. B Pattar, F. Restuccia, and T. Melodia, "DeepLoRa: Fingerprinting LoRa devices at scale through deep learning and data augmentation," in *Proc. ACM Int. Symposium Mob. Ad Hoc Netw. Comput. (MobiHoc)*, Shanghai, China, Jul. 2021.

[23] M. Piva, G. Maselli, and F. Restuccia, "The tags are alright: Robust large-scale RFID clone detection through federated data-augmented radio fingerprinting," in *Proc. ACM Int. Symposium Mob. Ad Hoc Netw. Comput. (MobiHoc)*, Shanghai, China, Jul. 2021.

[24] N. Soltani, K. Sankhe, J. Dy, S. Ioannidis, and K. Chowdhury, "More is better: Data augmentation for channel-resilient RF fingerprinting," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 66–72, 2020.

[25] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for RF device fingerprinting in cognitive communication networks," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 160–167, 2018.

[26] R. Das, A. Gadre, S. Zhang, S. Kumar, and J. M. Moura, "A deep learning approach to IoT authentication," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[27] A. Elmaghbub and B. Hamdaoui, "LoRa device fingerprinting in the wild: Disclosing RF data-driven fingerprint sensitivity to deployment variability," *IEEE Access*, vol. 9, pp. 142 893–142 909, 2021.

[28] R. Xie, W. Xu, Y. Chen, J. Yu, A. Hu, D. W. K. Ng, and A. L. Swindlehurst, "A generalizable model-and-data driven approach for open-set RFF authentication," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4435–4450, 2021.

[29] H. Ruotsalainen, G. Shen, J. Zhang, and R. Fujdiak, "LoRaWAN physical layer-based attacks and countermeasures, a review," *Sensors*, vol. 22, no. 9, p. 3127, 2022.

[30] G. Shen, J. Zhang, A. Marshall, M. Valkama, and J. Cavallaro, "Radio frequency fingerprint identification for security in low-cost IoT devices," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, 2021, pp. 309–313.

[31] ——, "Towards length-versatile and noise-robust radio frequency fingerprint identification," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2355–2367, 2023.

[32] S. Andrews, R. M. Gerdes, and M. Li, "Crowdsourced measurements for device fingerprinting," in *Proc. ACM Conf. Security Privacy Wireless Mobile Netw. (WiSec)*, 2019, pp. 72–82.

[33] K. Merchant and B. Nousain, "Enhanced RF fingerprinting for IoT devices with recurrent neural networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2019, pp. 590–597.

[34] S. D. Andrews, "Extensions to radio frequency fingerprinting," Ph.D. dissertation, Virginia Tech, 2019.

[35] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lille, France, July 2015, pp. 1180–1189.

[36] M. A. B. Temim, G. Ferré, B. Laporte-Fauret, D. Dallet, B. Minger, and L. Fuché, "An enhanced receiver to decode superposed LoRa-like signals," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7419–7431, 2020.

[37] C. Li, H. Guo, S. Tong, X. Zeng, Z. Cao, M. Zhang, Q. Yan, L. Xiao, J. Wang, and Y. Liu, "NELoRa: Towards ultra-low SNR LoRa communication with neural-enhanced demodulation," in *Proc. ACM Conf. Embedded Netw. Sensor Systems (SenSys)*, 2021, pp. 56–68.

[38] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A multi-channel software decoder for the LoRa modulation scheme," in *Proc. Int. Conf. Internet Things, Big Data Security (IoTBDS)*, Mar. 2018, pp. 41–51.

[39] W. C. Jakes and D. C. Cox, *Microwave mobile communications*. Wiley-IEEE press, 1994.

[40] S. Rajendran and Z. Sun, "RF impairment model-based IoT physical-layer identification for enhanced domain generalization," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1285–1299, 2022.

[41] K. Merchant and B. Nousain, "Toward receiver-agnostic RF fingerprint verification," in *Proc. IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.