SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2025, Vol. 68, Iss. 7, 170308:1–170308:16
https://doi.org/10.1007/s11432-024-4463-0

Special Topic: Integration of Large AI Model and 6G

Let RFF do the talking: large language model enabled lightweight RFFI for 6G edge intelligence†

Ning GAO[1], Yi LIU[1], Qifan ZHANG[1], Xiao LI[2] & Shi JIN[2*]

[1]*School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China*
[2]*National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China*

Received 1 November 2024/Revised 23 February 2025/Accepted 30 May 2025/Published online 23 June 2025

Abstract With the evolution of sixth-generation (6G) wireless networks, a large number of edge intelligent devices are now connected to the Internet of Things (IoT). Owing to the open nature of wireless networks, the massive edge IoT devices need to continuously prevent spoofing and the intrusion of malicious IoT devices. The deep learning (DL)-based radio frequency fingerprint identification (RFFI) provides a promising zero-trust edge IoT security scheme by automatically extracting the radio frequency fingerprint (RFF) feature from the intrinsic hardware imperfections. To address the problems in the training overhead, data limitation, and scalability for the current DL-based RFFI, we considered an outdoor long-range (LoRa) edge intelligent network, where we combined the large language model (LLM) for the first time and proposed a BERT-LightRFFI framework to enhance zero-trust edge IoT security. Specifically, we pre-trained a BERT model with the unlabeled data via self-supervised learning and obtained a powerful RFF feature extractor. Then, we used the knowledge distillation to inherit the BERT learn-gene to the small BERT-Light model and then fine-tuned a classifier of the pre-trained BERT-Light model by using few-shot labeled wireless data. The time complexity, parameter quantities, and computational complexity were analyzed. In the experiments, we used a large-scale real-world LoRa dataset to evaluate the performance of the proposed framework and suggest some interesting insights. The results prove the proposed framework's effectiveness, achieving an accuracy of 97.52% in the presence of multipath fading and Doppler shift, which is better than the previous benchmark methods.

Keywords radio frequency fingerprint identification (RFFI), large language model (LLM), knowledge distillation, zero-trust security, 6G edge intelligence

Citation Gao N, Liu Y, Zhang Q F, et al. Let RFF do the talking: large language model enabled lightweight RFFI for 6G edge intelligence. Sci China Inf Sci, 2025, 68(7): 170308, https://doi.org/10.1007/s11432-024-4463-0

1 Introduction

The next generation of wireless networks, i.e., the sixth-generation (6G), can provide a worldwide wireless coverage with exceptional speed, near-zero latency, and access by a massive number of devices, which is regarded as a promising technology that would bring people into the era of the Internet of Everything (IoE) [1]. According to a Cisco research report, the Internet of Things (IoT) had connected to 28 billion devices by the year 2022, and by 2030, it is estimated that the number will be 500 billion [2]. These connections have greatly promoted the development of edge intelligence, which can be found in areas such as health monitoring, intelligent driving, and smart homes. However, owing to the open nature of wireless networks, frequent access by massive edge IoT devices to the network can cause serious security risks. For instance, malicious IoT devices can access the network by monitoring and tampering with the legitimate media access control address and then launch a series of malicious attacks, i.e., a denial of service and man-in-the-middle attacks [3]. Although such threats can be mitigated by upper-layer encryption and authentication, but to a large extent, they cannot satisfy the requirements of edge intelligent applications for low latency and low power consumption [4].

Radio frequency fingerprint identification (RFFI) provides feasible zero-trust IoT security via a lightweight and low-latency approach, which provides an effective response to two adversarial scenarios: spoofing and reply attacks [5]. The radio frequency fingerprint (RFF) comes from the intrinsic hardware imperfections during the manufacturing process. As far as the radio frequency (RF) components

* Corresponding author (email: jinshi@seu.edu.cn)
† Special Topic: The 27th Annual Meeting of the China Association for Science and Technology

are concerned, RFF can be classified into frequency offset, inphase/quadrature (I/Q) imbalance, and power amplifier nonlinearity [6]. These unpredictable random variations make the RFF unique and unclonable, which thereby endows IoT devices access with security and reliability. RFFI can be formulated as a multi-class problem, where the standard RFFI process can be divided into three steps, namely, RFF pre-processing, classifier training, and classification. In recent years, with the development of deep learning (DL), the impact of DL on the wireless communications field is significant, and the RFFI field is no exception [7–9]. Compared with the traditional feature engineering-based RFFI, the DL-based RFFI can extract the fine-grained RFF feature automatically for identification without entirely considering the communication technology and protocol, thereby achieving great attention [10]. Nevertheless, most DL solutions in the wireless communications field are developed for specific tasks, and there have also been several problems with the DL-based RFFI. (1) The training overhead is large; it takes a considerable amount of time and computing power to train a high-performance deep neural network (DNN) for the RFFI. (2) The data are limited; in practice, most of the data acquired are inadequate and unlabeled, which cannot guarantee RFFI generalization. (3) The scalability is limited; the trained RFFI model is usually used for a specific classification task that requires a unique DNN structure.

Recently, the large language model (LLM), which is the foundational model built on natural language understanding, is regarded as the change maker for the next wave of artificial intelligence (AI) [11]. To name a few, OpenAI's ChatGPT uses the Transformer architecture with powerful natural language generation and understanding capabilities, and Google's BERT is a Transformer-based bidirectional encoder representation with rich language representations by pre-training on large-scale text data. The LLM is pre-trained on extremely large amounts of unlabeled data over millions of iterations using self-supervision techniques [12]. As a result, the LLM is highly generalizable and can be fine-tuned for specific tasks with few-shot or zero-shot learning, which has achieved significant success in terms of AI assistants, software development, and healthcare. Thanks to the LLM's versatile and powerful foundation for various tasks, researchers have gradually begun to explore the application of LLM in wireless communication fields [12–14]. However, the wireless LLM is still in its infancy, and how to apply the LLM to RFFI and the benefits that the LLM can bring to RFFI are still unknown. In this paper, we attempted to combine the LLM with the RFFI for the first time. Specifically, we considered an outdoor long-range (LoRa) edge intelligent network, where the RFFI model must be lightweight and the edge data inadequate and limitedly labeled. To deal with the above problems, we propose a BERT-enabled lightweight RFFI framework, namely, the BERT-LightRFFI framework, to enhance the zero-trust edge IoT security. The BERT-LightRFFI framework contains three models, which are the BERT model, the BERT-Light model, and the LightRFFI model. The main contributions of this paper are summarized as follows.

- As a first attempt, we combined the LLM with the RFFI and proposed a novel BERT-LightRFFI framework. Therein, the BERT model was pre-trained with the I/Q dataset via self-supervised contrastive learning; then, the knowledge distillation was used to inherit the BERT learn-gene to the BERT-Light model for high-performance RFF feature extraction. This proves the effectiveness of knowledge distillation in compressing the LLM and inheriting core knowledge.
- To match the wireless signal with the LLM and consider the model response time, we divided the I/Q data directly into the word vectors and extracted the semantic feature for BERT pre-training, which built a novel pre-processing path from wireless signals to semantics. To mitigate the influence of the channel effect on RFFI, i.e., the fading and Doppler shift, we used a cross-modal I/Q dataset to train the BERT model for the RFF feature extraction, where the unlabeled wired and wireless signals were used.
- The LightRFFI model constructed for the edge IoT devices is model-agnostic, requires no unique structure, and can adapt flexibly to various environments via a few-shot fine-tuning. The potential of the proposed BERT-LightRFFI was validated by using the real-world LoRa dataset, achieving an accuracy of 97.52% in the presence of multipath fading and Doppler shift, which is better than the previous state-of-the-art methods.

2 Related work

The RFFI has different categories of schedules. Based on the identification position of the signal segment, the RFFI can be classified into transient-based and steady-state-based methods. According to the transformation domain of the signal, the RFFI can be classified into time and transformation domains, i.e., the frequency and wavelet domains. As per the artificial feature injection, the RFFI can be divided

into passive identification and active identification [15]. In light of the feature extraction approach, the RFFI can be categorized as the feature engineering-based RFFI and the DL-based automatic extraction RFFI [16]. Here, we focused primarily on the last classification to discuss the related work. In the earlier study, the RFF was extracted by feature engineering, which depended on wireless expert knowledge. The widely used RFFs are the frequency offset, I/Q imbalance, power amplifier (PA) nonlinearity, phase errors, time-frequency spectrum, and the Hilbert–Huang spectrum [6]. Based on the feature engineering, identification performance can be further improved by machine learning such as random forest, or support vector machine. However, the feature engineering-based method usually requires accurate signal representation, which is difficult to achieve in complex communication scenarios. In the real world, different RFFs often couple with each other, and extracting one individual can compromise the correlation information between the various RFFs. Moreover, feature engineering usually takes a long time to conduct the feature extraction, which is a challenge for low latency and low power consumption scenarios.

Although training the DL-based RFFI model takes a considerable amount of time, identification with the trained model can typically be conducted in a few milliseconds. Meanwhile, the DL-based RFFI need not consider an accurate signal representation and the coupling relationship between different RFFs; thus, it has attracted extensive attention in recent years [17–19]. As one of the early studies, a convolutional neural network (CNN)-based RFFI has been proposed using the steady-state signal segment, which has shown that the DL-based approach outperforms the existing bispectrum and Hilbert-Huang spectrum-based methods [18]. Some subsequent studies are based on CNN, where Ref. [10] developed a spectrogram-CNN model to identify the 25 LoRa devices with an accuracy of 96.4%; however, it eliminates the channel effect by a wired attenuator. The authors in [19] proposed the AlexNet CNN architecture to explore the identification performance with the I/Q imbalance and the PA nonlinearity. Compared with the real-valued neural network, the complex-valued neural network (CVNN) can reserve the correlation between I/Q imbalance, which can bring about performance improvements [20–22]. Ref. [21] has shown the outstanding performance of CVNN for RFFI; however, these studies have high computing complexity or large model sizes. Model compression is a lightweight network technology that reduces computing complexity and model size, including knowledge distillation, neuron pruning, and quantization. The effectiveness of the model compression has been shown in different wireless communication applications, i.e., the CSI feedback and the semantic communications [23–25]. The authors in [26] have proposed a knowledge distillation-based CVNN compression to address the RFFI problem, which showed that there is almost no performance gap between the compressed CVNN and the original CVNN.

The channel effect is one of the crucial factors affecting RFFI accuracy, which results in the well-known problem of low RFFI accuracy in the "train on one day, test on another day" scenario. The authors in [27] showed that the wireless channel impacts identification accuracy significantly, i.e., from 85% to 9% and from 30% to 17% in the self-organizing dataset and the DARPA dataset, respectively. Some existing studies focus on eliminating the influence of the channel effect for RFFI [28–35]. Ref. [28] showed that the temporal variability of the wireless channel has a negative impact on the RFFI. A high-efficiency DeepFIR framework has been proposed to counteract the channel effect in wireless DL algorithms without retraining the underlying DL model [29]. The authors in [32] proposed a multisource feature fusion network to achieve excellent identification performance, where the original signal, the signal after demodulation, and the signal after channel equalization were considered; however, it was still sensitive to the various environments. To address the wireless channel impacts on the RFFI, the channel-independent features and the data augmentation have been investigated [34]. Furthermore, by adding an independent module at the front of the CNN classifier, the RFF filters have been studied to mitigate the channel effect; however, they introduce additional computational overhead and processing latency [33]. Few studies focus on the end-to-end channel effect mitigation for RFFI by using the LLM enabled lightweight network model. Therefore, in this paper, we attempt to mitigate the channel effect influence on the RFFI with the LLM and transfer the LLM learn-gene to the lightweight model for edge IoT devices.

3 System model

In this section, we first give the signal representation and then formulate the problem, the main symbols used in this paper are explained in Table 1.

Table 1 Symbol explanation.

Symbol	Meaning
Italic letter	Scalar
Bold lowercase letter	Vector
Bold uppercase letter	Matrix
$\lvert\cdot\rvert$	Vector size
$\mathcal{L}(\cdot)$	Loss function
$f(\cdot)$	Mapping function
$\log(\cdot)$	Logarithm
$\boldsymbol{A}^{\top}$	Matrix transpose
$\widetilde{\boldsymbol{x}}$	Estimation value of $\boldsymbol{x}$
$\hat{y}$	Prediction value of y

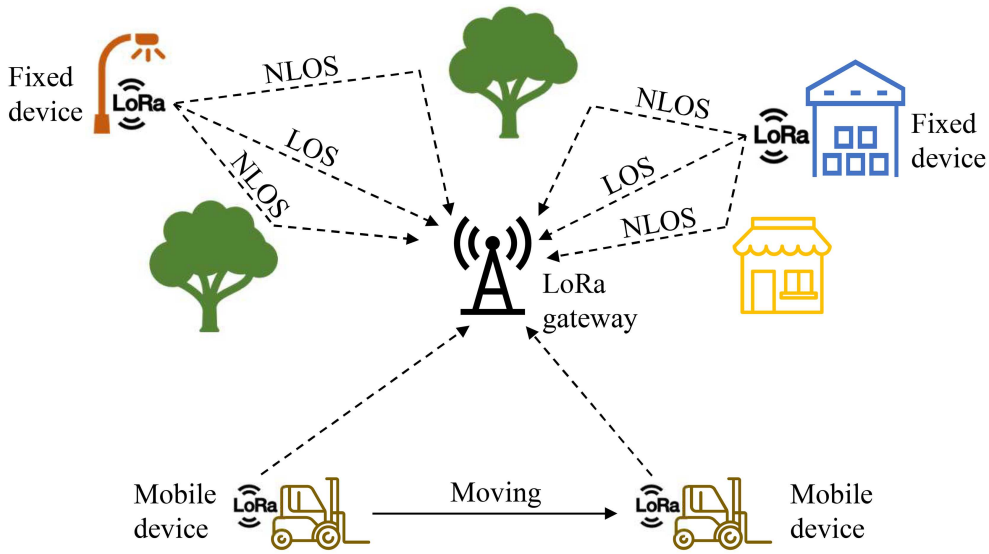

Figure 1 (Color online) Schematic of an outdoor star LoRa network.

3.1 Signal representation

As shown in Figure 1, we consider an outdoor LoRa network with a star topology that includes a series of fixed LoRa devices, such as various environmental sensors and a LoRa gateway that access the LoRa devices to the core networks, as well as mobile LoRa devices, each of which communicates wirelessly with the gateway.

Specifically, the LoRa employs the chirp spread spectrum (CSS) modulation technique, where the signal frequency linearly varies over time, the baseband frequency-modulated signal can be represented as [10]

$$s\left(t\right)=A\mathrm{e}^{\mathrm{j}(-\pi Bt+\pi\frac{B}{T}t^{2})}, \tag{1}$$

where A represents the amplitude, B represents the bandwidth, and T represents the duration. The duration can be expressed by using the spreading factor (SF) and the bandwidth, i.e.,

$$T=\frac{2^{\beta}}{B}, \tag{2}$$

where β indicates the number of the chip bits that can represent one information symbol. Since each signal component arrives at the receiver at different times, thus, with the superposition of the multipath fading, the received signal can be mathematically expressed as

$$r(t)\ =\ \sum_{i=0}^{N}\alpha_i(t)s_i(t-\tau_i(t)), \tag{3}$$

where N is the number of paths, $\alpha_i(t)$ is the attenuation coefficient of the ith path, $\tau_i(t)$ is the delay of the ith transmission path. With the LoRa device moving during signal transmission, the signal can

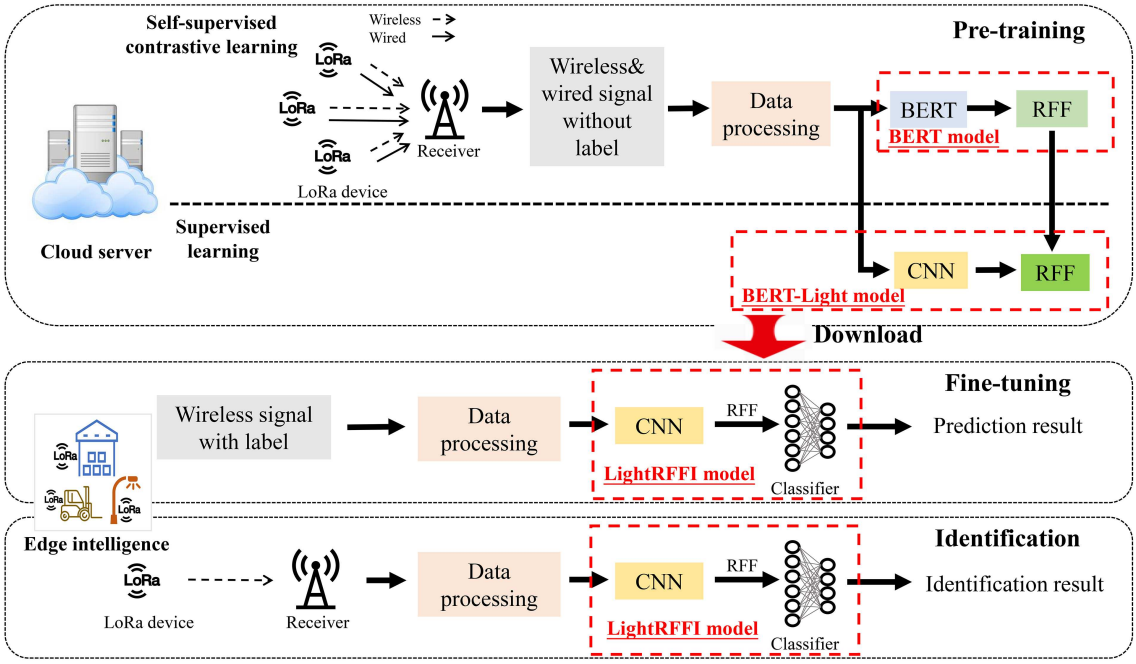

Figure 2 (Color online) Proposed BERT-LightRFFI framework.

experience a Doppler shift, which is given by

$$f' = \left(\frac{v \pm v_0}{v \mp v_s}\right) f, \tag{4}$$

where f' is the received frequency, f is the original frequency sent by the transmitter, v is the propagation speed of the signal in the current environment, v_0 is the moving speed of the receiver (if it is close to the transmitter, it is $+$, otherwise it is $-$), v_s is the moving speed of the transmitter (if it is close to the receiver, it is $-$, otherwise it is $+$).

In this case, the received signal can be expressed as

$$r(t) = h(t) * f_{\mathrm{RFF}}\big(s(t)\big) + n(t), \tag{5}$$

where $h(t)$ is the channel effect including the multipath fading and the Doppler shift, the symbol $*$ denotes convolution operation, the function $f_{\mathrm{RFF}}(\cdot)$ denotes the impact caused by the device hardware defects. The received signal will also introduce the RFF of the receiver, but the RFF is stable and unique due to only one receive gateway, so it will not disrupt the RF fingerprint feature of the transmitter and will not affect the identification, so the RFF of the receiver is not considered. The symbol $n(t)$ represents the additive white Gaussian noise (AWGN) within the wireless channel, which is stochastic and signal independent interference following a zero-mean complex Gaussian distribution with variance σ^2. In particular, the received signal can be rewritten as

$$r_{\mathrm{I}}(t) = r(t)\cos(2\pi f_c t), \tag{6}$$

and

$$r_{\mathrm{Q}}(t) = r(t)\sin(2\pi f_c t). \tag{7}$$

Therein, the symbols $r_{\mathrm{I}}(t)$ and $r_{\mathrm{Q}}(t)$ represent the I/Q signal, respectively, and f_c denotes the carrier frequency of the I/Q demodulation.

3.2 Problem formulation

The proposed BERT-LightRFFI framework is generally divided into three steps: the pre-training of the BERT model and the BERT-Light model, the fine-tuning of the LightRFFI model, and the deployment of the LightRFFI model, as shown in Figure 2. Considering the large amount of data and the training

overhead, the pre-training can be carried out by a professional on the cloud server. Due to the difficulty of obtaining enough labeled data in real-world, only a few-shot labeled wireless data are used for the LightRFFI fine-tuning. In the fine-tuning step, the individual user downloads the pre-trained BERT-Light model to local and fine-tunes its classifier, and then constructs the LightRFFI model. In the identification step, the well trained LightRFFI model is deployed on the edge IoT devices. When the edge IoT device receives the wireless signal, it processes the data and inputs it into the trained LightRFFI for RFF feature extraction and classification, thereby realizing the device identification.

We define $\boldsymbol{x}$ as a received signal, $\mathcal{X}$ ($\boldsymbol{x} \in \mathcal{X}$) as the sample space of all the received signal, y as the true device category corresponding to $\boldsymbol{x}$, and $\mathcal{Y}$ ($y \in \mathcal{Y}$) as the category space. Then, the dataset is represented by $\boldsymbol{D}_{\text{unlabeled}} = (\boldsymbol{x}_j, \widetilde{\boldsymbol{x}}_j)_{j=1}^{K}$ and $\boldsymbol{D}_{\text{labeled}} = (\widetilde{\boldsymbol{x}}_i, y_i)_{i=1}^{M}$, where $\boldsymbol{D}_{\text{unlabeled}}$ represents the dataset for pre-training stage, $\boldsymbol{x}_j$ represents the jth wired signal, $\widetilde{\boldsymbol{x}}_j$ represents the signal of $\boldsymbol{x}_j$ after data augmentation, and K represents the number of sample in $\boldsymbol{D}_{\text{unlabeled}}$. $\boldsymbol{D}_{\text{labeled}}$ represents the dataset in fine-tuning stage, $\widetilde{\boldsymbol{x}}_i$ represents the ith wireless signal, y_i represents the label corresponding to $\boldsymbol{x}_i$, and M represents the number of sample in $\boldsymbol{D}_{\text{labeled}}$ ($K \gg M$). In this case, the optimization problem of the pre-training stage can be represented as

$$\min_{\boldsymbol{W}_{\text{BERT}} \in \mathcal{W}} \mathbb{E}_{(\boldsymbol{x}, \widetilde{\boldsymbol{x}}) \sim \boldsymbol{D}_{\text{unlabeled}}} \mathcal{L}_{\text{SCL}}(f_{\text{BERT}}(\boldsymbol{x}; \boldsymbol{W}_{\text{BERT}}),\ f_{\text{BERT}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERT}})),\ \boldsymbol{W}_{\text{BERT}} : \boldsymbol{W}_{s1} \to \boldsymbol{W}_{\text{op1}}, \quad (8)$$

and

$$\begin{aligned} &\min_{\boldsymbol{W}_{\text{BERTLight}} \in \mathcal{W}} \mathbb{E}_{(\widetilde{\boldsymbol{x}}) \sim \boldsymbol{D}_{\text{unlabeled}}} \mathcal{L}_{\text{MSE}}(f_{\text{BERTLight}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERTLight}}),\ f_{\text{BERT}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERT}})), \\ &\boldsymbol{W}_{\text{BERTLight}} : \boldsymbol{W}_{s2} \to \boldsymbol{W}_{\text{op2}}, \end{aligned} \quad (9)$$

where $\mathcal{W}$ is the parameter set, $\boldsymbol{W}_{s1}$ and $\boldsymbol{W}_{s2}$ represents the initial parameters of BERT and BERTLight respectively, $\boldsymbol{W}_{\text{op1}}$ and $\boldsymbol{W}_{\text{op2}}$ represents the optimal parameters of BERT and BERTLight, respectively, $\mathcal{L}_{\text{SCL}}$ and $\mathcal{L}_{\text{MSE}}$ represents the supervised contrastive loss (SCL) and the mean squared error (MSE), respectively, and $f_{\text{BERT}}(\cdot)$ and $f_{\text{BERTLight}}(\cdot)$ represents the mapping of the feature extractor to the data, respectively. Meanwhile, the optimization problem of the fine-tuning stage can be expressed as

$$\min_{\boldsymbol{W}_c \in \mathcal{W}} \mathbb{E}_{(\widetilde{\boldsymbol{x}}, y) \sim \boldsymbol{D}_{\text{labeled}}} \mathcal{L}_{\text{CE}}(f_{\text{classifier}}(f_{\text{BERTLight}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERTLight}}); \boldsymbol{W}_c),\ y),\ \boldsymbol{W}_c : \boldsymbol{W}_{s3} \to \boldsymbol{W}_{\text{op3}}, \quad (10)$$

where $\boldsymbol{W}_c$ is the parameters of the classifier, $\boldsymbol{W}_{s3}$ represents the initial parameters of the classifier in the fine-tuning stage, $\boldsymbol{W}_{\text{op3}}$ represents the optimal parameters of the model in this stage, $\mathcal{L}_{\text{CE}}$ represents the loss function in fine-tuning stage, constraining the model prediction results to be close to y, and $f_{\text{classifier}}(\cdot)$ represents the mapping of the classifier to the data.

4 Methods

In this section, we first introduce the architectures of the model in the proposed framework, give the methods of the data processing and the embedding, then explain the details of pre-training and fine-tuning, and finally analyze the complexity of the proposed framework.

4.1 Model architecture

The proposed framework contains three models, one is an LLM, namely the BERT model, which is trained using self-supervised contrastive learning to extract the fine-grained feature, and the other two are the BERT-Light model and the LightRFFI model, which are trained using the supervised learning.

BERT. The BERT model is based on the classic BERT model, which is an LLM containing 6 Transformer encoders and a multi-head attention mechanism with 8 attention heads, as shown in Figure 3. The specific calculation process of the model is that the input data query $\boldsymbol{Q}$, key $\boldsymbol{K}$, and value $\boldsymbol{V}$ are first transformed three times linearly through the fully connected layer, and mapped to a matrix of word vectors. Then, the three matrices are used to calculate the attention score, which can be written as [36]

$$\boldsymbol{A}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d_s}}\right)\boldsymbol{V}, \quad (11)$$

where $\boldsymbol{A}(\cdot)$ is the attention score operation, d_s represents the vector dimension corresponding to each attention head. In our model settings, the value of d_s is 64, so that each self-attention module can learn

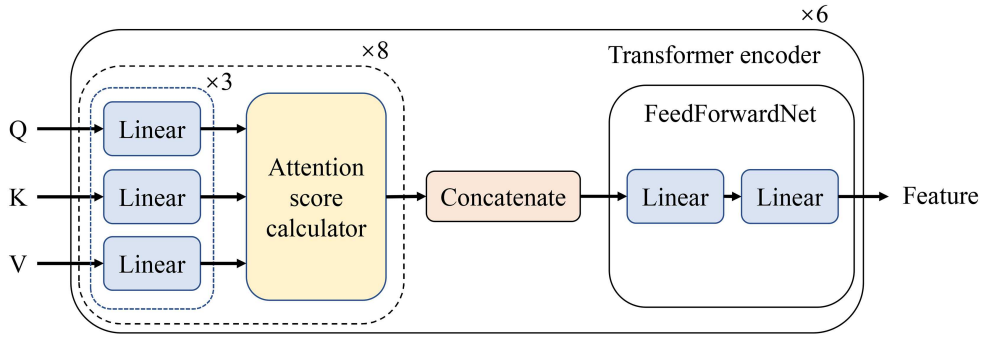

Figure 3 (Color online) BERT model architecture based on the classic BERT model.

different semantic features of the input data. The matrices calculated by the eight attention heads are concatenated to restore the shape to the same shape as the original input, and finally the calculation is completed through the fully connected layer. The multi-head attention mechanism enhances the model's ability to capture semantic information from different representation subspaces and splits the data into multiple attention heads for calculation, which improves the stability of training and reduces the risk of overfitting.

Next, the original data is added to the output of the multi-head attention to obtain an effective feature representation containing the relationship between the words, which fully explores the impact of the RFF on different locations in the wireless signal. Finally, the input data passes through a feedforward neural network consisting of two fully connected layers and a ReLU activation function. The output and input form a residual connection, and the output is normalized. The feedforward neural network introduces nonlinear changes to improve the expressiveness of the model and further learn the feature representation. Through the above architecture, the model can directly capture the global dependencies between any positions, avoiding the shortcomings of many current deep learning methods that focus on local features and have difficulty capturing long-distance dependencies. Therefore, it is more robust to environmental changes and has better generalization capabilities.

BERT-Light and LightRFFI. Comparing the two models, the LightRFFI model has only one more classifier than the BERT-Light model. Due to the unique residual connection structure, the ResNet has an excellent feature extraction capability with a small number of convolutional layers, which is suitable for deploying on the edge IoT device[1)]. Therefore, the LightRFFI model deployed on the edge IoT device is implemented based on the ResNet and a classifier, which includes 8 convolutional layers, 1 pooling layer, and 1 fully connected layer. The specific structure is shown in Figure 4. All convolutional layers are 1-dimensional to adapt to the input of wireless data. Specifically, the first layer of the model uses a convolutional layer with a convolution kernel size of 1×7, followed by a ReLU activation function, a batch normalization layer, and a maximum pooling layer of 2×2. The model is subsequently composed of 4 residual blocks, each of which includes 1 convolutional layer with a convolution kernel size of 1×3 and a stride of 2, which are also followed by an ReLU activation function and a batch normalization layer. The input and output form a residual connection. In particular, before the final output of the 2nd, 3rd, and 4th residual blocks, they are downsampled by a convolutional layer with a convolution kernel size of 1×1 and a stride of 2. The final data are downsampled three times, and the output data length is reduced from 4096 to 512.

4.2 Data processing

Since the received signal is in a complex form, the data slicing and data augmentation are required before the model training.

Data slicing. The signal received by the receiver has a large number of sampling points, which is difficult to calculate directly for the model, so it is necessary to slice the data to speed up the training. We set the window size to 4096 sampling points, the step size is the same as the window size, and the signal is divided into a series of slices without overlapping. In the dataset, the wired signal of each device is divided into 400 slices, and a total of 25 devices generate a total of 10000 data fragments.

1) The BERT-Light in the proposed framework includes but not limited to the ResNet; many classic CNN models and self-constructed models can also work, i.e., the VGG, the MobileNet, and the GoogLeNet, etc.

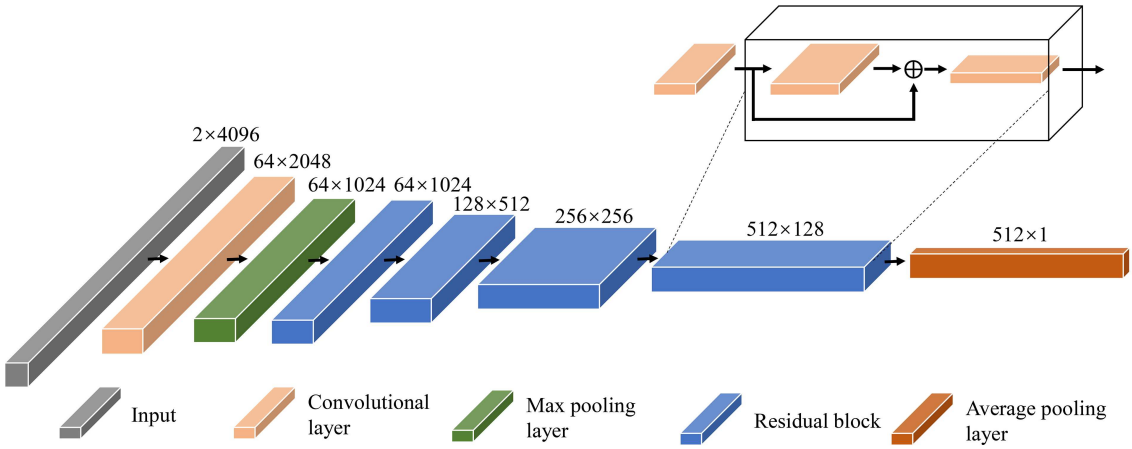

Figure 4 (Color online) BERT-Light architecture based on the ResNet.

Algorithm 1 Data processing procedure.

Input: Wired signal $\boldsymbol{x}_{\text{origin}}$, window size w;
Output: Wired signal slice $\boldsymbol{x}$ and wireless signal slice $\widetilde{\boldsymbol{x}}$ that match the model input;
Initialize the channel model, set maximum number, start index st and end index ed = st + w − 1;
for i = 1 to maximum number **do**
  Take out the signal slice in the window: $\boldsymbol{x}_i = \boldsymbol{x}_{\text{origin}}[\text{st} : \text{ed}]$;
  Move the window backward with a step size of w: st = st + w, ed = ed + w;
  Take out the real part $\boldsymbol{x}_{\text{I}}^i$ and imaginary part $\boldsymbol{x}_{\text{Q}}^i$ of the signal slice, separately;
  The signal slice $\boldsymbol{x}_i$ passes through the wireless channel and add noise, get $\widetilde{\boldsymbol{x}}_i$;
  Take out the real $\widetilde{\boldsymbol{x}}_{\text{I}}^i$ part and imaginary part $\widetilde{\boldsymbol{x}}_{\text{Q}}^i$ of the signal slice, separately;
end for
Cross-permutate each element in the real and imaginary parts of the wired signal slice $\boldsymbol{x}$ and wireless signal slice $\widetilde{\boldsymbol{x}}$ as input to the BERT model;
Concatenate the real and imaginary parts of the wireless signal slice $\widetilde{\boldsymbol{x}}$ as input to the BERT-Light model.

Data augmentation. Data augmentation completes the task of generating a rich wireless signal by passing the wired signal through the wireless channel with different channel parameters. For the data augmentation, it should be ensured that the wired signal and the wireless signal are essentially the same, and there is only a difference in the channel effect. The goal is to ensure that the data alignment problem cannot affect the model training, which is conducive to the model learning channel-independent RFF feature. The simulation of the wireless channel can be implemented using Matlab.

The input data of the BERT are wired and wireless signals, which are divided into two parts, the real part I and the imaginary part Q, with a length of 4096. Each element in the two parts is cross-arranged to form new data in the shape of IQIQIQ$\cdots$, with a length of 8192. Using sequence data composed of alternating I/Q pairs can avoid information loss caused by I/Q separation, which is conducive to the LLM model learning the coupling relationship between them. At the same time, it simulates the characteristics of natural language and regards an I/Q pair as a minimum unit of language, so as to better adapt to the model architecture and improve the model's ability to capture signal features. In addition, directly using I/Q pairs reduces the overhead caused by the domain transformation of the signal in the air interface, thereby improving the system response speed. In this case, the wired signal and the wireless signal are used as two independent inputs of the BERT model, and the data shape is [batchsize, 8192]. On the hand, the wireless signal is used as the input data of the BERT-Light model, which is also divided into the real part I and the imaginary part Q, but it should be pointed out that the I/Q pairs are not cross-arranged and directly input to the BERT-Light model as two channels, which further compresses the processing time. Thus, the shape of the input data is [batchsize, 2, 4096]. The details are shown in Algorithm 1.

4.3 Embedding

As mentioned in Subsection 4.1, the Transformer encoder calculates the relationship between the words and the input data, so its input data should be in a form similar to natural language, but the received signal after data processing is still a string of numbers and cannot be used directly for calculation. Thus, before inputting the data into the Transformer encoder, it should be embedded.

The input signal data of the LLM are sliced into signal segments of 8192 sampling points. We take every 128 sampling points as a word and regard the segment as a whole sentence of 64 words. The classification (CLS) tag is inserted at the beginning of the sentence as the vector representation of the entire sentence. The CLS tag is initially set to all zeros and the length is the same as that of a single word, so a sentence of 65 words is finally obtained. After the word segmentation operation is completed, we use a single fully connected layer to map each word to a vector of length 512. The fully connected layer is followed by a ReLU activation function to introduce nonlinearity and improve the representation ability of the word vector. Embedding is the first part of the model and is continuously optimized through backward propagation during the training process. The learnable word embedding operation helps the model capture complex language patterns and the key connections between words.

Particularly, compared with the embedding operation of the classic BERT, the used BERT model discards the two operations of segment embedding and position embedding. The reasons are that segment embedding is utilized to distinguish different sentences in the input, but our input only has one sentence, so sentence embedding is not required, and solubility enhancing peptide (SEP) tags are not used for sentence spacing. Meanwhile, position embedding introduces additional word position information to the input data, which can help the model understand the order relationship between words. However, the hardware imperfections mainly come from defects in the manufacturing process, which can be regarded as stable for a long time after the device is manufactured. Furthermore, the temperature and humidity of the environment do not change significantly over a period of time, which guarantees the RFF stability. Therefore, the order of the impact on the data in the time dimension can be ignored for our considered scenario.

4.4 Pre-training and fine-tuning

The purpose of the pre-training phase is to train the BERT model for RFF feature extraction and inherit its learn-gene to the BERT-Light model. First, the BERT model is trained to obtain a powerful RFF feature extractor, and then the BERT-Light model is trained to obtain a lightweight RFF feature extractor. Then, based on the pre-trained BERT-Light model, the fine-tuning is implemented by freezing the BERT-Light model parameters and training a classifier with a few-shot labeled wireless data.

BERT training. Considering the data limited labeled, the BERT model is first trained through unlabeled self-supervised contrastive learning to obtain a powerful RFF feature extractor. Specifically, the information noise contrastive estimation (infoNCE) loss is used, which is calculated by using cosine similarity and can be written as

$$\mathcal{S}_c = \frac{\boldsymbol{F} \cdot \widetilde{\boldsymbol{F}}}{\max(||\boldsymbol{F}||_2, \epsilon) \cdot \max(||\widetilde{\boldsymbol{F}}||_2, \epsilon)}, \tag{12}$$

where $\boldsymbol{F}$ and $\widetilde{\boldsymbol{F}}$ represent the two data for the cosine similarity calculation, ϵ is a very small positive number, here it is 1E$-$8 to avoid division by zero. Before calculating cosine similarity, the data are normalized, and mathematically expressed as

$$\boldsymbol{F} = \frac{\boldsymbol{F}}{\max(||\boldsymbol{F}||_2, \epsilon)}, \tag{13}$$

and

$$\widetilde{\boldsymbol{F}} = \frac{\widetilde{\boldsymbol{F}}}{\max(||\widetilde{\boldsymbol{F}}||_2, \epsilon)}. \tag{14}$$

Therefore, the loss function $\mathcal{L}_{\mathrm{SCL}}$ can be written as

$$\mathcal{L}_{\mathrm{SCL}} = -\frac{1}{K} \sum_{i=1}^{K} \log \frac{\exp(\mathcal{S}_c^{ii}/\mathcal{T})}{\sum_{j=1}^{K} \exp(\mathcal{S}_c^{ij}/\mathcal{T})}, \tag{15}$$

where K represents the number of samples, $\mathcal{T}$ is the temperature scaling factor used to control the sharpness of the cosine similarity $\mathcal{S}_c$, $\mathcal{S}_c^{ij}$ represents the cosine similarity between the ith data and the jth data. When i is not equal to j, it represents the cosine similarity of the negative sample pair, and when i is equal to j, it represents the cosine similarity between the positive samples.

Specifically, the feature extracted by the model is first normalized, then the cosine similarity is calculated, and then temperature scaling is performed. Finally, the feature similarity between positive samples is maximized and the feature similarity between negative samples is minimized based on the cross entropy loss. In this case, the model can learn to shorten the distance between the wired signal and wireless signal feature of the same device. As a result, amplifying the distance between the signal feature of different devices, and promoting the model to extract RFF feature that is independent of the channel effect. Therein, the positive sample pair is the wired data and its corresponding augmented data, and the negative sample pair is the other data in the same batch.

BERT-Light training. The BERT-Light model is trained by knowledge distillation under the supervision of the RFF feature extracted by the BERT. The goal is to guide the BERT-Light model to achieve an RFF feature extraction capability as excellent as the BERT model. Meanwhile, it generates a small model to increase the inference speed and reduce the computational overhead. The loss function is based on the MSE loss. The MSE loss can be given by [37, 38]

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{K}\sum_{i=1}^{K}\left(\boldsymbol{F}^{i}_{\mathrm{BERTLight}} - \boldsymbol{F}^{i}_{\mathrm{BERT}}\right)^{2}, \tag{16}$$

where the symbol $\boldsymbol{F}^{i}_{\mathrm{BERTLight}}$ is the ith RFF feature of the BERT-Light, and $\boldsymbol{F}^{i}_{\mathrm{BERT}}$ is the ith RFF feature of the BERT. The MSE loss $\mathcal{L}_{\mathrm{MSE}}$ optimizes the model so that the distance of the feature extracted by the BERT-Light model and the BERT model is as small as possible. Thus, the BERT-Light model can extract features related to the relationship between the words without using the self-attention mechanism. It is worth mentioning that the input of the model is universal and does not target any signal form specific to any communication protocol, so it can be applied to many wireless networks such as the LoRa and the WiFi.

LightRFFI fine-tuning. The classifier is implemented using a single fully connected layer, with an input feature dimension of 512 and an output dimension of 25[2)]. The cross entropy loss is used as the loss function, which is expressed as

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{L} y_{ij}\log\left(\hat{y}_{ij}\right), \tag{17}$$

where M is the number of samples in fine-tuning, $|\mathcal{Y}| = L$ is the number of classifications, y_{ij} denotes the true label of the ith sample belonging to the jth classification, while $\hat{y}_{ij}$ denotes the probability that the network predicts that the ith sample belongs to the jth classification.

Due to the difficulty in obtaining the adequate labeled dataset in real-world environments, the labeled data used for the classifier training only account for 20% of the total dataset. The LightRFFI model is trained with a few-shot labeled wireless data of specific wireless environments, which are in accordance with the edge intelligence scenarios. Thanks to the easily distinguishable feature extracted by the BERT model, the BERT-Light model also achieves high-quality feature extraction in supervised learning training. Thus, fine-tuning the LightRFFI model with a small amount of labeled data can achieve excellent performance. The complete training process is presented in Algorithm 2.

4.5 Complexity analysis

We first analyze the time complexity of the BERT-LightRFFI framework by splitting it into two parts. The time complexity of the BERT model mainly comes from the embedding operation and the six Transformer encoders. The embedding operation is completed by the fully connected layer, and the time complexity can be expressed as $\mathcal{O}(\ell \times d_{\mathrm{ori}} \times d)$, where ℓ represents the length of the input sequence, d_{ori} represents the length of the original word, and d represents the length of the word vector after mapping. The complexity of the Transformer encoder mainly comes from the matrix operation between linear transformation and query $\boldsymbol{Q}$, key $\boldsymbol{K}$, and value $\boldsymbol{V}$. The time complexity of linear transformation can be expressed as $\mathcal{O}(\ell \times d^2)$; the time complexity of matrix operations can be expressed as $\mathcal{O}(\ell^2 \times d)$. Therefore, the overall time complexity of the Transformer encoder can be expressed as

$$\mathcal{O}(\ell \times d^2 + \ell^2 \times d). \tag{18}$$

2) Here, the output dimension can be arbitrarily set according to the specific task of the edge IoT device.

Algorithm 2 Pre-training and fine-tuning process.

Require: Wired signal $\boldsymbol{x}$, wireless signal $\widetilde{\boldsymbol{x}}$, and label y;
PRE-TRAINING BERT&BERT-Light:
Initialize: Set learning rate η, maximum epochs;
Training on the dataset $\boldsymbol{D}_{\text{unlabeled}}$:
for epoch = 1 to maximum epochs **do**
  for $\forall\{\boldsymbol{x}, \widetilde{\boldsymbol{x}}\}$ in $\boldsymbol{D}_{\text{unlabeled}}$ **do**
    Embedding data;
    Forward propagation: $\boldsymbol{F}_{\text{BERT}} = f_{\text{BERT}}(\boldsymbol{x}; \boldsymbol{W}_{\text{BERT}})$; $\widetilde{\boldsymbol{F}}_{\text{BERT}} = f_{\text{BERT}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERT}})$;
    Calculate the loss: $\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{SCL}}(\boldsymbol{F}_{\text{BERT}}, \widetilde{\boldsymbol{F}}_{\text{BERT}})$ as (15);
    Backward propagation: Update model weights $\boldsymbol{W}_{\text{BERT}}$ by Adam, $\boldsymbol{W}_{\text{BERT}} \leftarrow \boldsymbol{W}_{\text{BERT}} - \eta \frac{\partial \mathcal{L}_{\text{BERT}}}{\partial \boldsymbol{W}_{\text{BERT}}}$;
  end for
end for
for epoch = 1 to maximum epochs **do**
  for $\forall\{\widetilde{\boldsymbol{x}}\}$ in $\boldsymbol{D}_{\text{unlabeled}}$ **do**
    Load and freeze BERT parameters $\boldsymbol{W}_{\text{BERT}}$;
    Forward propagation: $\widetilde{\boldsymbol{F}}_{\text{BERTLight}} = f_{\text{BERTLight}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERTLight}})$; $\widetilde{\boldsymbol{F}}_{\text{BERT}} = f_{\text{BERT}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERT}})$;
    Calculate the loss: $\mathcal{L}_{\text{BERTLight}} = \mathcal{L}_{\text{MSE}}(\widetilde{\boldsymbol{F}}_{\text{BERTLight}}, \widetilde{\boldsymbol{F}}_{\text{BERT}})$ as (16);
    Backward propagation: Update model weights $\boldsymbol{W}_{\text{BERTLight}}$ by Adam, $\boldsymbol{W}_{\text{BERTLight}} \leftarrow \boldsymbol{W}_{\text{BERTLight}} - \eta \frac{\partial \mathcal{L}_{\text{BERTLight}}}{\partial \boldsymbol{W}_{\text{BERTLight}}}$.
  end for
end for
FINE-TUNING LightRFFI:
Initialize: Set learning rate η, maximum epochs;
Training on $\boldsymbol{D}_{\text{labeled}}$:
for epoch = 1 to maximum epochs **do**
  Load and freeze BERT parameters $\boldsymbol{W}_{\text{BERT}}$ and BERT-Light parameters $\boldsymbol{W}_{\text{BERTLight}}$;
  for $\forall\{\widetilde{\boldsymbol{x}}, y\}$ in $\boldsymbol{D}_{\text{labeled}}$ **do**
    Forward propagation: $\hat{y} = f_{\text{Classifier}}(f_{\text{BERTLight}}(\widetilde{\boldsymbol{x}}; \boldsymbol{W}_{\text{BERTLight}}); \boldsymbol{W}_c)$;
    Calculate the loss: $\mathcal{L}_c = \mathcal{L}_{\text{CE}}(\hat{y}, y)$ as (17);
    Backward propagation: Update model weights $\boldsymbol{W}_c$ by Adam, $\boldsymbol{W}_c \leftarrow \boldsymbol{W}_c - \eta \frac{\partial \mathcal{L}_c}{\partial \boldsymbol{W}_c}$;
  end for
end for

Table 2 Model parameters and calculation amount.

Model	Total params	Params size (MB)	Estimated total size (MB)	MFLOPs
BERT-RFFI	18968064	72.36	14156.85	1231.09
BERT-LightRFFI	705280 (↓ 96.3%)	2.69 (↓ 96.3%)	13.73 (↓ 99.9%)	133.5 (↓ 89.2%)

The overall time complexity of the BERT model can be given by

$$\mathcal{O}(\ell \times d_{\text{ori}} \times d) + \mathcal{O}(\ell \times d^2 + \ell^2 \times d). \tag{19}$$

The time complexity of the BERT-Light model and LightRFFI model are mainly composed of convolutional layers, two pooling layers, and a fully connected layer. We assume that the input sample length is $|\boldsymbol{x}| = V$, the convolution kernel size is K_{conv}, the max pooling layer window size is $K_{\max}$, the average pooling layer window size is K_{avg}, the number of input channels is C_{in}, the number of output channels is C_{out}, and the number of the convolutional layer is U. Then, the time complexity of a convolutional layer is $\mathcal{O}(C_{\text{in}} \times C_{\text{out}} \times K_{\text{conv}} \times V)$, the max pooling layer is $\mathcal{O}(C_{\text{in}} \times K_{\max} \times V)$, the average pooling layer is $\mathcal{O}(C_{\text{in}} \times K_{\text{avg}} \times V)$, and the fully connected layer is $\mathcal{O}(F_{\text{in}} \times F_{\text{out}})$, where F_{in} is the number of input feature and F_{out} is the number of output feature. The overall time complexity can be defined as

$$\mathcal{O}\left(\sum_{u=1}^{U} C_{\text{in}}^u \times C_{\text{out}}^u \times K_{\text{conv}}^u \times V\right) + \mathcal{O}\left(C_{\text{in}} \times K_{\max} \times V\right) + \mathcal{O}\left(C_{\text{in}} \times K_{\text{avg}} \times V\right) + \mathcal{O}\left(F_{\text{in}} \times F_{\text{out}}\right). \tag{20}$$

The parameter quantities and computational complexity of the BERT-LightRFFI framework are calculated via a quantitative analysis, which is shown in Table 2. We can find that the total parameter number has decreased by 96.3%, the parameter size has reduced by 96.3%, the estimated total size has reduced by 99.9% and the mega floating-point operations per second (MFLOPs) has decreased by 89.2%, respectively.

Table 3 Training parameters.

Parameter	Pre-training	Fine-tuning
Batchsize	128	32
Learning rate	0.0001	0.0001
Maximum epoches	200	100
Number of data in each category	400	80
Data size	1×8192	2×4096

5 Experimental results

5.1 Experimental setup

All the experiments are carried out on a server configured with NVIDIA Tesla V100 (32 GB) GPU, Intel Xeon E5-2698 CPU, and 512 GB of RAM. We construct the DNN based on the PyTorch framework. The wired dataset used is from the publicly available LoRa dataset collected by Elmaghbub et al. [5], which includes the time-domain I/Q signal along with their frequency-domain signal through the fast Fourier transform (FFT). Specifically, the datasets are collected using an IoT device testing platform. The testing platform consists of 25 identical Pycom IoT devices and a USRP B210 receiver, which operates at a center frequency of 915 MHz with a sampling rate of 1 MS/s. The dataset encompasses a variety of scenarios, including wired, wireless, indoor, and outdoor environments. In particular, the wired datasets are collected over five days and each device performs 10 transmissions per day. To ensure the fairness of the dataset, we select five transmissions per device per day to train and test the model. We use 80% of the entire data as a training set, 10% as a validation set, and 10% as a test set. The training parameters are shown in Table 3.

5.2 Performance metrics

The confusion matrix is an important method in machine learning for evaluating the performance of classification, which is often used in RFFI. It is an $n \times n$ matrix, where n is the number of classifications. The sum of each row in the confusion matrix represents the actual sample count for the classification represented by that row, while the sum of each column represents the predicted sample count for the classification represented by that column. The diagonal elements of the matrix represent the samples that are predicted correctly, indicating that the predicted classification matches the true classification. On the other hand, the accuracy represents the proportion of correctly classified samples among all the predicted samples, and it is one of the most intuitive and commonly used evaluation metrics for the RFFI. The calculation involves dividing the number of correct classifications by the total number of classifications.

5.3 Feasibility analysis

To explore the RFFI performance of the BERT model, we train the same classifier according to the method in Subsection 4.4 and obtain the BERT-RFFI model. The confusion matrices of the training results of the BERT-RFFI model and the LightRFFI model are shown in Figure 5, respectively. Comparing Figures 5(a) and (b), it can be seen that the BERT-RFFI model and the LightRFFI model can achieve an excellent performance. It is interesting to find that the performance of the LightRFFI model even exceeds that of the BERT-RFFI model with a few-shot fine-tuning. Figure 5(c) is the accuracy of a LightRFFI model without the knowledge distillation (LightRFFI-NKD). Compared Figure 5(b) with Figure 5(c), it can be observed that without relying on the knowledge of the BERT model, it is difficult to effectively extract RFF features from the wireless data. This suggests that the knowledge distillation achieves effective learn-gene transfer, and the RFF feature representation of the BERT model is obtained with the architecture of the ResNet, which is an effective model compression.

To further analyze the insightful mechanism of the BERT-LightRFFI framework, we visualize the high-dimensional RFF feature using the t-distributed stochastic neighbor embedding (t-SNE) technique and then explore the RFF feature variation. Figure 6(a) is the feature visualization of the LightRFFI-NKD model, which shows that there are no clear boundaries between the 25 categories and they are almost completely overlapped together. In Figure 6(b), by a few-shot fine-tuning, the RFF features gradually become clear between 25 categories, but a few-shot sample training is not enough for the LightRFFI-NKD model to obtain a satisfactory performance. From Figure 6(c), we can observe that the boundary between

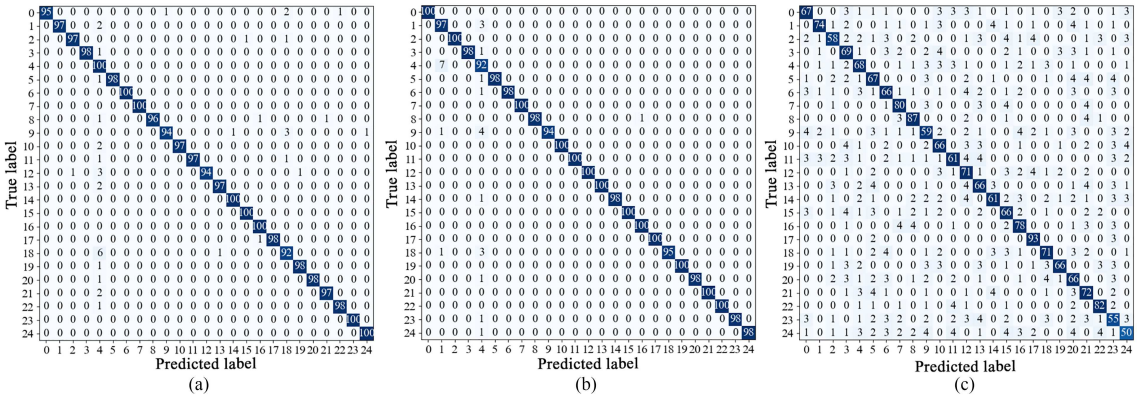

Figure 5 (Color online) Confusion matrix of model. (a) BERT-RFFI, ACC = 96.89%; (b) LightRFFI, ACC = 97.52%; (c) LightRFFI-NKD, ACC = 76.57%.

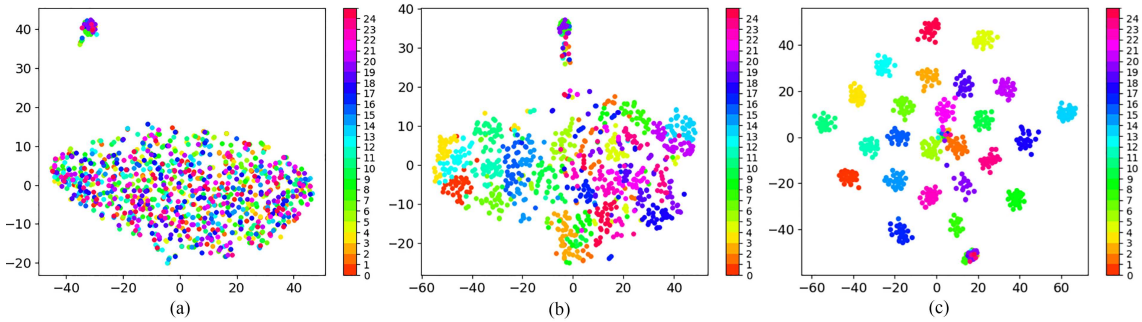

Figure 6 (Color online) Feature visualization. (a) LightRFFI-NKD model without a few-shot training; (b) LightRFFI-NKD model with a few-shot training; (c) LightRFFI model.

Table 4 Datasets under different channel environments.

Parameter	Value
Number of samples	4096
SNR	{5, 10, 15, 20, 25, 30}
Path delays	{{0, 1E−6, 2E−6}, {0, 1E−6, 2E−6, 3E−6}, {0, 1E−6, 2E−6, 3E−6, 4E−6}}
Average path gains	{{0, −3, −5}, {0, −3, −5, −7}, {0, −3, −5, −7, −10}}
Doppler shift	{0, 5, 10, 15, 20, 25}

the features of each category is relatively clear for the LightRFFI model, which also explains the reason of 97.52% accuracy in Figure 5(b).

5.4 Impact of channel effect

The multipath fading and Doppler shift on the signal are described in (3) and (4). To more intuitively see the impact of the channel effect on the signal, we take a fragment of the wired signal from the dataset and pass it through a channel simulation with five propagation paths. The generated wireless signal is compared with the original signal as shown in Figure 7(a). Next, we use the same fragment of the wired signal and pass it through a simulated channel with a 20 Hz Doppler shift and five propagation paths. The generated wireless signal is compared with the original signal, which is shown in Figure 7(b). The results show that the multipath fading seriously distorts the signal waveform, and this phenomenon is more obvious when the Doppler shift is added. In order to explore the impact of different channel effects on model performance, we generate multiple datasets through simulation, and the specific details are shown in Table 4.

The accuracies of the LightRFFI model in different channel environments are shown in Figure 8. According to the results shown in Figure 8(a), different SNRs can have a great impact on the accuracy of the model. It can be found that the LightRFFI model is difficult to perform an accurate RFFI in a low SNR environment, but in the medium and high SNR environments, the LightRFFI model can achieve

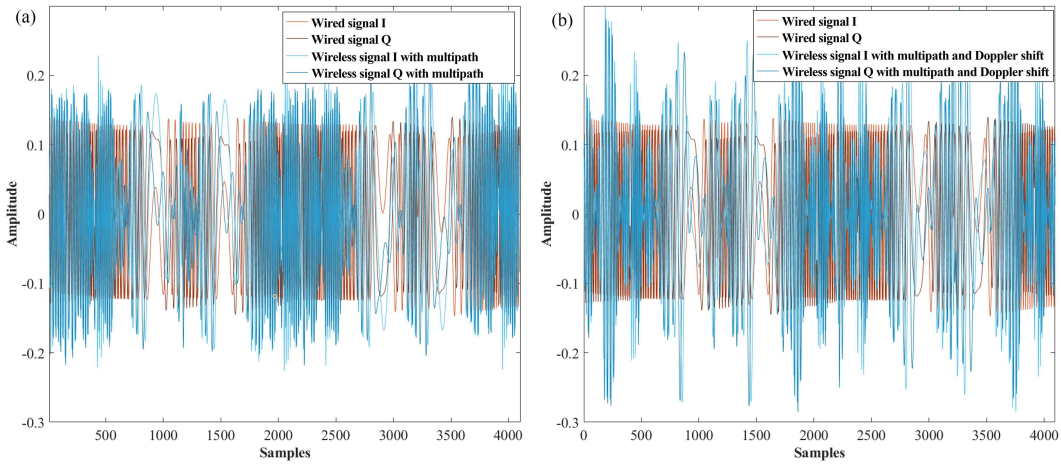

Figure 7 (Color online) Wireless signals vs. wired signals. (a) Wireless signals containing only multipath fading; (b) wireless signals including the multipath fading and the Doppler shift.

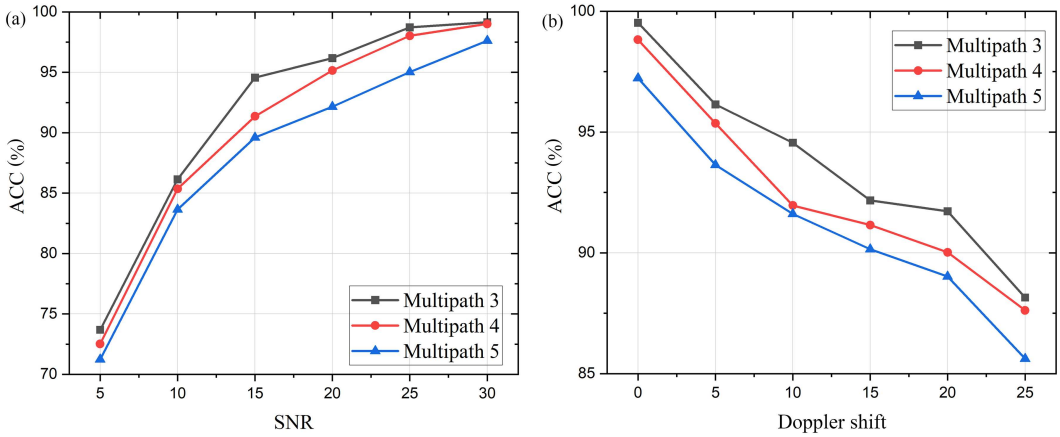

Figure 8 (Color online) Accuracy of the BERT-Light in different channel environments. (a) Multipath with different SNRs; (b) multipath with different Doppler shifts.

extremely high identification accuracy. As per the results in Figure 8(b), we can see that the difficulty of identifying a device in a mobile environment is higher than that in a stationary wireless environment, but in most of the outdoor low speed scenarios, the accuracy of the LightRFFI model can be maintained at a relatively high level. In addition, both Figures 8(a) and (b) show the influence of multipath fading on accuracy. As the number of propagation paths increases, the signal distortion becomes increasingly serious, and the difficulty of RFF feature extraction increases, resulting in a certain decrease in accuracy.

5.5 Comparison of different models

In this section, we compare the proposed framework with the recent state-of-the-art methods that apply contrastive learning and few-shot learning, including SA2SEI [39], CVNN [26], and MAT-CL [40]. Specifically, SA2SEI uses a novel adversarial enhancement-driven self-supervised learning and knowledge transfer framework, and fine-tunes the feature extractor and classifier using a small amount of labeled data. CVNN uses complex-valued neural networks for feature extraction and knowledge distillation for model compression. MAT-CL introduces pseudo labels in metric learning for semi-supervised metric learning, and uses an objective function regularized alternately by SSML and virtual adversarial training (VAT) to extract generalized semantic features of wireless signal. The accuracy of the four methods can be found in Table 5.

Compared with the three benchmark methods, our proposed LightRFFI model outperforms the other three methods in terms of parameter quantity and accuracy. This is attributed to the fine grained RFF feature extracted by the BERT model and the knowledge distillation to inherit the BERT learn-gene to the BERT-Light model. The extracted features of the three benchmark methods are visualized after

Table 5 Model parameters and calculation amount. The best results are in bold.

Model	Total params	MFLOPs	Accuracy (%)
SA2SEI [39]	1908864	138.2	94.16
CVNN [26]	1071367	**114.63**	95.4
MAT-CL [40]	859274	206.79	92.81
LightRFFI	**705280**	133.5	**97.52**

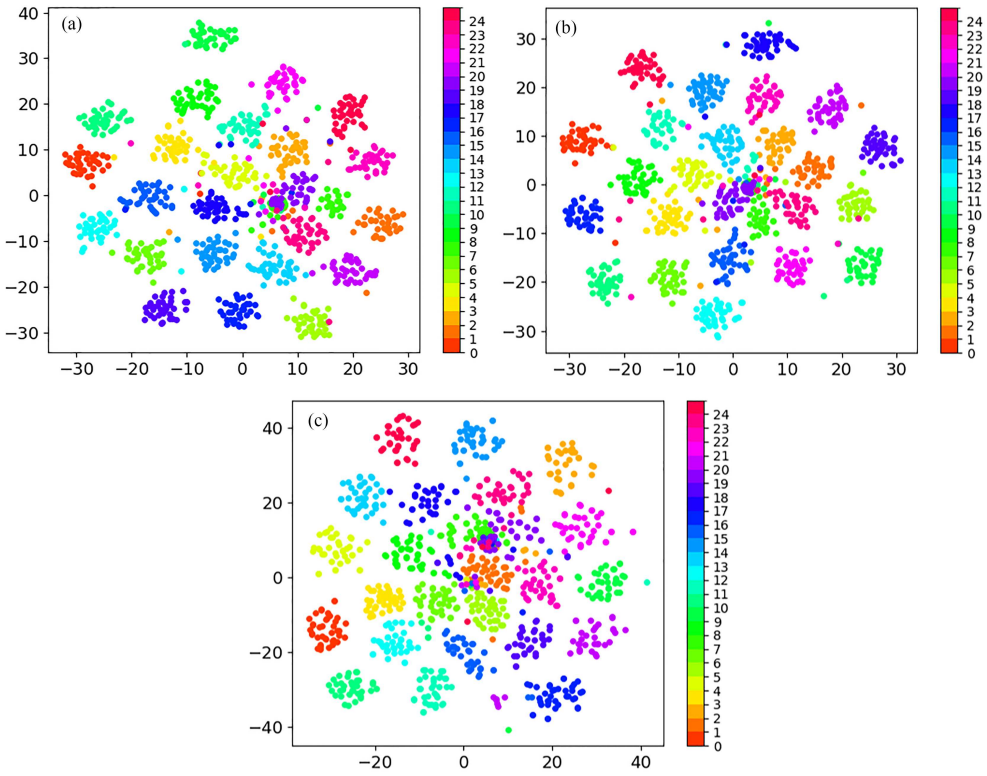

Figure 9 (Color online) Feature visualization. (a) SA2SEI; (b) CVNN; (c) MAT-CL.

t-SNE, which is shown in Figure 9. Compared Figures 9(a)–(c) with Figure 6(c), the results show that the boundary between the feature of each category is the clearest for the proposed LightRFFI model, almost equivalent for the SA2SEI and the CVNN, and the worst for the MAT-CL.

6 Conclusion

In this paper, we combined the LLM and proposed a BERT-LightRFFI framework to enhance zero-trust edge IoT security. Specifically, we considered an outdoor LoRa edge intelligent network, where the RFFI model can be deployed on edge IoT devices with limited resources. We pre-trained a BERT model with the unlabeled data via self-supervised learning and obtained a powerful RFF feature extractor. Subsequently, we used the knowledge distillation to inherit the BERT learn-gene to the BERT-Light model for the lightweight, and then fine-tuned the BERT-Light model and a classifier using a few-shot labeled wireless data. The time complexity, parameter quantities, and computational complexity were analyzed. In the experiments, we used a large-scale real world LoRa dataset to evaluate the performance of the proposed framework and found the interesting insights. The results proved the proposed framework's effectiveness in LLM compression and core knowledge inheritance, achieving an accuracy of 97.52% in the presence of multipath fading and Doppler shift, which is better than the previous benchmark methods. It found that in most of the outdoor low speed scenarios, the accuracy of the proposed LightRFFI model can be maintained at a relatively high level. Moreover, it was found that the accuracy of the LightRFFI model even exceeds that of the BERT-RFFI model with a few-shot fine-tuning.

Acknowledgements This work was supported in part by National Key Research and Development Program of China (Grant No.

2024YFE0200700), National Natural Science Foundation of China (Grant No. 62371131), and Program of Zhishan Young Scholar of Southeast University (Grant No. 2242024RCB0030).

References

1 Matthaiou M, Yurduseven O, Ngo H Q, et al. The road to 6G: ten physical layer challenges for communications engineers. IEEE Commun Mag, 2021, 59: 64–69
2 Zikria Y B, Ali R, Afzal M K, et al. Next-generation Internet of Things (IoT): opportunities, challenges, and solutions. Sensors, 2021, 21: 1174
3 Alipour H, Al-Nashif Y B, Satam P, et al. Wireless anomaly detection based on IEEE 802.11 behavior analysis. IEEE Trans Inform Forensic Secur, 2015, 10: 2158–2170
4 Xu D, Li T, Li Y, et al. Edge intelligence: empowering intelligence to the edge of network. Proc IEEE, 2021, 109: 1778–1837
5 Elmaghbub A, Hamdaoui B. Domain-agnostic hardware fingerprinting-based device identifier for zero-trust IoT security. IEEE Wireless Commun, 2024, 31: 42–48
6 Xu Q, Zheng R, Saad W, et al. Device fingerprinting in wireless networks: challenges and opportunities. IEEE Commun Surv Tut, 2016, 18: 94–104
7 Zuo Y, Guo J, Gao N, et al. A survey of blockchain and artificial intelligence for 6G wireless communications. IEEE Commun Surv Tut, 2023, 25: 2494–2528
8 Gao N, Qin Z, Jing X, et al. Anti-intelligent UAV jamming strategy via deep Q-networks. IEEE Trans Commun, 2020, 68: 569–581
9 Fang H, Wang X, Hanzo L. Learning-aided physical layer authentication as an intelligent process. IEEE Trans Commun, 2019, 67: 2260–2273
10 Shen G, Zhang J, Marshall A, et al. Radio frequency fingerprint identification for LoRa using deep learning. IEEE J Sel Areas Commun, 2021, 39: 2604–2616
11 Wu S, Fei H, Qu L, et al. NExt-GPT: any-to-any multimodal LLM. In: Proceedings of the 41st International Conference on Machine Learning, 2024
12 Zhang R, Du H, Liu Y, et al. Interactive AI with retrieval-augmented generation for next generation networking. IEEE Netw, 2024, 38: 414–424
13 Zhang R, Du H, Liu Y, et al. Generative AI agents with large language model for satellite networks via a mixture of experts transmission. IEEE J Sel Areas Commun, 2024, 42: 3581–3596
14 Zhou H, Hu C, Yuan Y, et al. Large language model (LLM) for telecommunications: a comprehensive survey on principles, key techniques, and opportunities. IEEE Commun Surv Tut, 2024. doi: 10.1109/COMST.2024.3465447
15 Gao N, Meng S, Li C, et al. RIS-assisted wireless link signatures for specific emitter identification. IEEE Trans Wireless Commun, 2024, 23: 17872–17883
16 Qi X, Hu A, Chen T. Lightweight radio frequency fingerprint identification scheme for V2X based on temporal correlation. IEEE Trans Inform Forensic Secur, 2024, 19: 1056–1070
17 Yu J, Hu A, Li G, et al. A robust RF fingerprinting approach using multisampling convolutional neural network. IEEE Internet Things J, 2019, 6: 6786–6799
18 Ding L, Wang S, Wang F, et al. Specific emitter identification via convolutional neural networks. IEEE Commun Lett, 2018, 22: 2591–2594
19 Zhang J, Woods R, Sandell M, et al. Radio frequency fingerprint identification for narrowband systems, modelling and classification. IEEE Trans Inform Forensic Secur, 2021, 16: 3974–3987
20 Tu Y, Lin Y, Hou C, et al. Complex-valued networks for automatic modulation classification. IEEE Trans Veh Technol, 2020, 69: 10085–10089
21 Gopalakrishnan S, Cekic M, Madhow U. Robust wireless fingerprinting via complex-valued neural networks. In: Proceedings of IEEE Global Communications Conference, 2019. 1–6
22 Fan Z, Tu Y, Lin Y, et al. Class-incremental learning for recognition of complex-valued signals. IEEE Trans Cogn Commun Netw, 2024, 10: 417–428
23 Guo J, Wang J, Wen C K, et al. Compression and acceleration of neural networks for communications. IEEE Wireless Commun, 2020, 27: 110–117
24 Xie H, Qin Z. A lite distributed semantic communication system for Internet of Things. IEEE J Sel Areas Commun, 2020, 39: 142–153
25 Lin Y, Tu Y, Dou Z. An improved neural network pruning technology for automatic modulation classification in edge devices. IEEE Trans Veh Technol, 2020, 69: 5703–5706
26 Wang Y, Gui G, Gacanin H, et al. An efficient specific emitter identification method based on complex-valued neural networks and network compression. IEEE J Sel Areas Commun, 2021, 39: 2305–2317
27 Al-Shawabka A, Restuccia F, D'Oro S, et al. Exposing the fingerprint: dissecting the impact of the wireless channel on radio fingerprinting. In: Proceedings of IEEE Conference on Computer Communications, 2020. 646–655
28 Saeif A, Savio S, Gabriele O. The day-after-tomorrow: on the performance of radio fingerprinting over time. In: Proceedings of Annual Computer Security Applications Conference, 2023. 439–450
29 Restuccia F, D'Oro S, Al-Shawabka A, et al. DeepFIR: channel-robust physical-layer deep learning through adaptive waveform filtering. IEEE Trans Wireless Commun, 2021, 20: 8054–8066
30 Soltani N, Sankhe K, Dy J, et al. More is better: data augmentation for channel-resilient RF fingerprinting. IEEE Commun Mag, 2020, 58: 66–72
31 Wang N, Li W, Jiao L, et al. Orientation and channel-independent RF fingerprinting for 5G IEEE 802.11ad devices. IEEE Internet Things J, 2022, 9: 9036–9048
32 Zhang Y, Zhang Q, Zhao H, et al. Multisource heterogeneous specific emitter identification using attention mechanism-based RFF fusion method. IEEE Trans Inform Forensic Secur, 2024, 19: 2639–2650
33 Fu H, Peng L, Liu M, et al. Deep learning-based RF fingerprint identification with channel effects mitigation. IEEE Open J Commun Soc, 2023, 4: 1668–1681
34 Shen G, Zhang J, Marshall A, et al. Towards scalable and channel-robust radio frequency fingerprint identification for LoRa. IEEE Trans Inform Forensic Secur, 2022, 17: 774–787
35 Liu Y, Gao N, Chen Y, et al. Lightweight RF fingerprint identification using cross modal knowledge distillation: learning from wired to wireless. In: Proceedings of IEEE International Conference on Wireless Communications and Signal Processing, 2024. 554–559
36 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Conference on Neural Information Processing Systems, 2017. 6000–6010
37 Romero A, Ballas N, Kahou S E, et al. FitNets: hints for thin deep nets. 2015. ArXiv:1412.6550
38 Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. 2015. ArXiv:1503.02531
39 Liu C, Fu X, Wang Y, et al. Overcoming data limitations: a few-shot specific emitter identification method using self-supervised learning and adversarial augmentation. IEEE Trans Inform Forensic Secur, 2024, 19: 500–513
40 Fu X, Peng Y, Liu Y, et al. Semi-supervised specific emitter identification method using metric-adversarial training. IEEE Internet Things J, 2023, 10: 10778–10789