GPU-Free Specific Emitter Identification Using Signal Feature Embedded Broad Learning

Yibin Zhang, *Student Member, IEEE*, Yang Peng, Jinlong Sun, *Member, IEEE*, Guan Gui, *Senior Member, IEEE*, Yun Lin, *Member, IEEE*, and Shiwen Mao, *Fellow, IEEE*

***Abstract*—Emerging wireless networks may suffer severe security threats due to the ubiquitous access of massive wireless devices. Specific emitter identification (SEI) is considered as one of the important techniques to protect wireless networks, which aims to identifying legal or illegal devices through the radio frequency (RF) fingerprints contained in RF signals. Existing SEI methods are implemented with either traditional machine learning or deep learning. The former relies on manual feature extraction which is usually inefficient, while the latter relies on the powerful graphics processing unit (GPU) computing power but with limited applications and high cost. To solve these problems, in this article, we propose a GPU-free SEI method using a signal feature embedded broad learning network (SFEBLN), for efficient emitter identification based on a single-layer forward propagation network on the central processing unit (CPU) platform. With this method, the original RF data is first preprocessed through external signal processing nodes, and then processed to generate mapped feature nodes and enhancement nodes by nonlinear transformation. Next, we design the internal signal processing nodes to extract effective features from the processed RF signals. The final input layer consists of mapped feature nodes, enhancement nodes, and internal signal processing nodes. Then, the network weight parameters are obtained by solving the pseudo inverse problem. Experiments are conducted over the CPU platform and the results show that our proposed SEI method using SFEBLN achieves a superior identification performance and robustness under various scenarios.**

***Index Terms*—GPU-free, radio frequency (RF) signals, signal feature embedded broad learning network (SFEBLN), specific emitter identification (SEI).**

Manuscript received 13 January 2023; revised 23 February 2023; accepted 12 March 2023. Date of publication 15 March 2023; date of current version 7 July 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0113003; in part by the Jiangsu Provincial Key Research and Development Program under Grant BE2020084-5; in part by the National Natural Science Foundation of China under Grant 61901228; and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX22-0948 and Grant KYCX21-0736. *(Corresponding author: Guan Gui.)*

Yibin Zhang, Yang Peng, Jinlong Sun, and Guan Gui are with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 2021010208@njupt.edu.cn; 2020010210@njupt.edu.cn; sunjinlong@njupt.edu.cn; guiguan@njupt.edu.cn).

Yun Lin is with the College of Information and Communication Engineering, Harbin Engineering University, Harbin 150009, China (e-mail: linyun@hrbeu.edu.cn).

Shiwen Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: smao@ieee.org).

Digital Object Identifier 10.1109/JIOT.2023.3257479

I. Introduction

With the rapid development of wireless communications, existing wireless networks are able to support the access of massive user terminals, including many wireless edge devices. At the network layer and data link layer, device identification usually depends on the Internet protocol (IP) address or media access control (MAC) address identifier. However, the IP address is not unique, and the MAC address is easy to be tampered with, which provide attack opportunities to hackers. Hence, to further improve the security of wireless communications, many specific emitter identification (SEI) [1], [2], [3] methods based on radio frequency fingerprinting (RFF) have been proposed. Thanks to the reliability and uniqueness of RFF, SEI can ensure the wireless communications security for Internet of Things (IoT) for smart homes [4], intelligent vehicle networks [5], Industrial IoT (IIoT) communications [6], [7], etc.

In recent years, some scholars discovered the radio frequency (RF) defects of signal transmitters. Scanlon et al. [8] attempted to extract a unique identifier from wireless signal transmissions in order to perform automated device identification. From the unique characteristics of the electromagnetic wave emitted by the transmitter, the RFF is unique [9]. The typical structure of wireless digital transmitter and its corresponding receiver is shown in Fig. 1. The RF signals are usually generated from baseband signals through raw inphase/quadrature (I/Q) digital modulation. The I/Q signals are going through digital/analog (D/A) modules, intermediate frequency (IF), up-converters, and a public nonlinear amplifier, respectively. The D/A module will introduce quantization error and nonlinear integral effect, and the IF will generate filter noise. In addition, the up-converter utilizing an oscillator will lead to frequency offset and the amplifier is usually nonlinear. Hence, the RF signals generated by different transmitters contain their own hardware features, which are termed as RFF.

In the past decade, artificial intelligence (AI), including traditional machine learning (ML) and deep learning (DL), have achieved great success in the field of signal processing, e.g., for automatic modulation recognition [10], [11], [12], [13], channel state information prediction and feedback [14], [15], fast beamforming design [16], intrusion detection [17], malware traffic classification [18], flight delay prediction [19], indoor localization [20], and other applications. However, these technologies based on ML or DL all need the support of a graphics

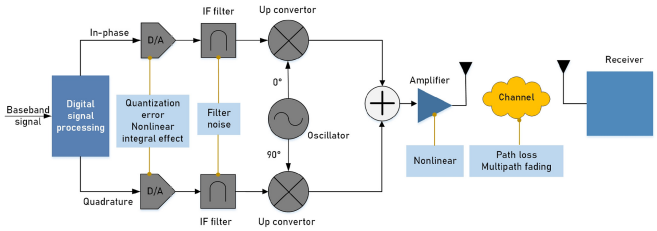


Fig. 1. Illustration of the structure of wireless digital transmitter and its corresponding receiver.

processing unit (GPU) for its huge computing power, which limits their use in practice. Hence, this article proposes to leverage the novel broad learning system (BLS) proposed in [21], [22], [23], and [24]. The BLS is a single-layer forward propagation network, which does not need back propagation to adjust the network weight. Furthermore, the BLS is capable of fast and efficient training due to its special architecture, which provides a promising solution for online learning tasks.

Broad learning has been widely applied in signal processing, e.g., for image recognition [25], Internet of Vehicles [26], [27], fault diagnosis [28], etc. The core advantage of broad learning is to save computing overheads. In other words, it is a lightweight solution. For the IoT system composed of edge devices with low computational power, lightweight design is a way to provide real-time solutions [30], [31]. In this article, we focus on the BLS and apply broad learning to SEI, aiming to develop efficient methods with limited computing overhead. The main contributions of this article are highlighted as follows.

1) We propose a GPU-free SEI method using the signal feature embedded broad learning network (SFEBLN). The proposed method utilizes massive nonlinear transformation nodes to approximate the fingerprinting features of RF signals. In addition, all the computation of the proposed SFEBLN are excuted on the central processing unit (CPU) platform.
2) According to the characteristics of RF signals, we design external and internal signal processing nodes to preprocess the original signals and extract statistical features. The processed signal features are embedded into the broad learning network for achieving a better SEI performance.
3) We conduct experiments to validate the proposed SFEBLN method on large-scale real-world open data sets of source automatic dependent surveillance-broadcast (ADS-B) signals [29]. The comparison results of computing overhead prove the feasibility of the proposed SFEBLN method for edge devices and online learning. Furthermore, the simulation results show that the proposed SFEBLN method achieves the state-of-the-art identification performance.

The remainder of this article is organized as follows. In Section II, we give a survey about existing SEI researches in recent years, including ML-based, DL-based, and BLS-based SEI methods. Section III shows the system model and the corresponding mathematical problem model. In Section IV, we discuss our proposed SEI method in detail, including basic knowledge, the network architecture, and the algorithm flow chart. The experimental results and some comparative analysis are given in Section V. Finally, Section VI conclude this article.

II. Related Works

In this section, we review the existing SEI methods in recent years, including the conventional methods based on combination of signal features and ML, the methods based on DL, and the latest methods based on BLS. Some of the algorithms described in this section are adopted in our subsequent comparative experiments as benchmark schemes.

A. ML-Based SEI Methods

The ML-based methods have been developed in early years. For example, Bertoncini et al. [32] proposed three methods to extract RFF for RFID labels, called dynamic wavelet fingerprint, wavelet packet decomposition, and higher order statistics, respectively. Then the authors utilized four conventional classification methods to verify the performance, which are linear and quadratic discriminant classifiers, *k*-nearest neighbor (kNN), and support vector machine (SVM). Huang et al. [33] proposed to utilize the permutation entropy of received signals as RF fingerprints, and used the kNN as the classifier. The authors verified their methods with steady-state signals from four wireless network cards and three digital radios. Zhang et al. [34] proposed to rely on the energy entropy, and first- and second-order moments of the Hilbert spectrum as RF fingerprints. Meanwhile, this article adopts SVM as the classifier. Satija et al. [35] explored variational mode decomposition (VMD) to decompose RF signals into specific numbers as RF fingerprints. The kNN was used to complete the SEI task. Gok et al. [36] employed the VMD to real radar signals and also achieved a great success.

B. DL-Based SEI Methods

In recent years, DL has achieved great success in computer vision (CV). Therefore, many scholars try to apply CV algorithms to the field of signal processing. For example, Merchant et al. [37] designed a convolutional neural network (CNN) to detect physical features for identification of cognitive radio devices. And, the method achieved great identification accuracy and robustness. Yin et al. [38] applied the RF fingerprint identification technique to LTE terminal identification, and proposed a multichannel CNN to extract the differential constellation trace figures to identify LTE terminals. Ramasubramanian et al. [39] claimed that the I/Q signals showed a "helical" structure and encoded the intrinsic signal features into short-term variations. Then, a 3-D CNN was used to extract the short-term space-time characteristics of I/Q signals. Qian et al. [40] proposed a multilevel sparse representation-based identification algorithm for SEI, which combined neural networks with sparse representation-based classification. In the multiscale CNN, the channel attention mechanism was introduced to extract low-dimensional and high-dimensional signal features. Du et al. [41] introduced a

variable network architecture search (NAS) mechanism for SEI to automatically search for the optimal models.

Furthermore, there are researches proposed that the RF signals should be preprocessed for adaptation to the DL-based algorithms. They coincidentally convert RF signals into picture signals. Lin et al. [42] designed a framework to transform complex-valued signals into the so-called contour stellar image, which extracted deep statistical information from raw wireless signals and presented it in the form of images. Peng et al. [43] employed heat constellation trace figures to replace the I/Q samples, which reduced the cost of training and avoided complex feature extraction. A slice integration cooperation method constellation are also proposed to improve the accuracy of RFF identification.

The above researches proved that DL-based algorithms are able to achieve good performance in SEI tasks. Therefore, more and more researches are carried out with DL-based SEI methods. For example, incremental learning (IL) strategy, narrowband system, channel robust problem, and so on. He and Wang [44] proposed cooperative identification by multiple distorted receivers, which achieved high diversity gain in the identification performance. In addition, they considered multiple fingerprints to simulate a practical emitter. Xiao and Yan [45] utilized the short-time Fourier transform (STFT) and k-means algorithm to obtain the time-frequency spectrograms of radar emitter signals. And, then used the CNN for emitter identification based on the time-frequency images. Liu et al. [46] explored the application of IL in noncryptographic device identification. To better understand the internal mechanisms of DNNs, the degree of conflict and conflict of fingerprints were proposed to analyze DNN models. Finally, the enhanced channel separation-enabled IL strategy was proposed for wireless device identification. Zhang et al. [47] comprehensively modeled the impairments of transmitter and receiver in narrowband systems, including oscillator imperfections, phase and gain imbalances of the mixer, and power amplifier nonlinearity. In addition, the authors further explored their effects on RF fingerprint identification and proposed a CNN-based RFFI protocol. Yang et al. [48] proposed a data-independent RFF extraction scheme using random data segments. The proposed LAFS scheme computed converged tap coefficients as RFF by minimizing the divergence between the desired signal and the demodulated reference signal. Shen et al. [49] proposed a deep metric learning-based scalable RFF identification framework, which enabled devices flexibly join and leave RFF database. Furthermore, a novel channel-independent spectrogram and data augmentation were utilized to achieve channel robustness. Chatterjee et al. proposed a deep neural network-based system extremely robust and secure at low cost, called RF-PUF [50]. The key idea of RF-PUF was based on the physical unclonable functions (PUFs). Furthermore, RF-PUF made lightweight design for wireless nodes to save costs as much as possible.

C. BLS-Based SEI Methods

BLS is a relatively new network architecture, and some scholars have recently applied it to SEI, for example, Xu et al. [51] proposed a lightweight SEI method called adaptive broad learning network (ADBLN), which incorporates an adaptive node expansion strategy to adjust the network structure according to RF signals. The proposed ADBLN is evaluated using the real world data collected from aircrafts and achieved a great identification performance.

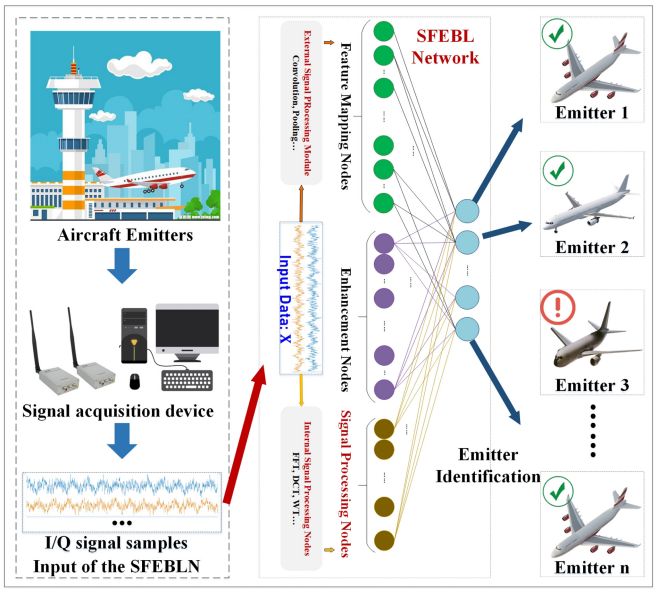


Fig. 2. Illustration of the rapid authentication system of civil aircrafts.

III. System Model and Problem Formulation

A. System Model

Wireless devices will be affected by many factors in the process of production. Each device in different transmitters has a certain device tolerance, or hardware defect [52]. Considering the transmitter structure shown in Fig. 1 and the multiple effects of transmitter hardware defects, the RF fingerprint can be modeled as the following:

$$\mathrm{RFF}(\cdot) = A_0^{\gamma}(1 + h_{PA}(\tilde{a}_{tx})) \times \exp\{i\theta + h_t(\sigma_{TIE}^m) + h_{\Delta}(\Delta_n, \Delta_{INL}) + h_m(\xi) + h_p(\rho_h, \rho_v)\} \quad (1)$$

where A_0^{γ} and θ are amplitude and phase information of received signal, respectively. Furthermore, $h_{PA}(\tilde{a}_{tx})$, $h_t(\sigma_{\mathrm{TIE}}^m)$, $h_{\Delta}(\Delta_n, \Delta_{\mathrm{INL}})$, $h_m(\xi)$, and $h_p(\rho_h, \rho_v)$ are nonlinear parameters introduced by amplifier, digital signal processing, DAC, mixer, and antenna, respectively.

Without loss of generality, this article focuses on the rapid authentication of civil aircraft based on ADS-B signals (a representative SEI application), aiming to quickly detect illegal aircraft [29]. As shown in Fig. 2, there are three parts of the authentication system: 1) data acquisition; 2) SFEBLN; and 3) aircraft identification. First, the aircrafts are considered as specific emitters in the system and the signal acquisition device is deployed near the airport. Then, the received ADS-B signals are the input of SFEBLN, which includes external and internal signal processing nodes. Finally, the SFEBLN is able to identify different aircrafts and discover unauthorized

illegal aircraft. The SFEBLN architecture shown in the figure is only a schematic diagram. The feature mapping nodes, enhancement nodes, and signal processing nodes are explained in detail in the following text. Meanwhile, the output nodes can be dynamically adjusted according to different tasks.

B. Problem Formulation

This article aims to propose a fast and efficient SEI method to identify the legitimacy of aircrafts. We utilize ADS-B signals collected at the airport in this study. The received RF signals can be expressed as follows:

$$\boldsymbol{x}_i(t) = \boldsymbol{s}_i(t) * \boldsymbol{h}_i(t) + n_i(t), \;\; i = 1, 2, \ldots, N \tag{2}$$

where $\boldsymbol{x}_i(t)$ is the received time series signal, $\boldsymbol{s}_i(t)$ and $\boldsymbol{h}_i(t)$ denote the transmitted ADS-B signal and the propagation channel, respectively, and $n_i(t)$ represents the additive noise.

Actually, the SEI is to solve the matching problem between the received RF signals and the corresponding aircraft labels. Assume that the data set is defined as $\mathbb{D}\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{y}_i$ is the label of the corresponding aircraft. Then, the problem to be solved can be defined as follows:

$$\widehat{\boldsymbol{y}_i} = f_{\mathrm{SEI}}\big(\boldsymbol{x}_i(t), \widetilde{\mathbf{W}}\big), i = 1, 2, \ldots, N \tag{3}$$

where $f_{\mathrm{SEI}}(\cdot)$ represents the mapping function used to identify the received signals, $\widehat{\boldsymbol{y}_i}$ is the predict aircraft label of $\boldsymbol{x}_i$, and $\widetilde{\mathbf{W}}$ denotes the optimal weight coefficient. We define the probability of a group of signals being identified correctly as the accuracy rate, given by

$$\alpha = \left\{ \sum_{i=1}^{N} \big(\widehat{\boldsymbol{y}_i} = \boldsymbol{y}_i\big) \right\} / N \tag{4}$$

and the corresponding error rate is $\epsilon = 1 - \alpha$. Therefore, we have the objective function to be optimized, that is, for the highest accuracy and the lowest error, i.e.,

$$\widetilde{\mathbf{W}} = \arg\min_{\mathcal{W}} \quad \epsilon\big\{\widehat{\boldsymbol{y}_i} = f_{\mathrm{SEI}}(\boldsymbol{x}_i, \mathcal{W})\boldsymbol{y}_i\big\} \tag{5}$$

where the optimal weight $\widetilde{\mathbf{W}}$ and the framework of $f_{\mathrm{SEI}}(\cdot)$ are the solutions to the SEI problem.

IV. Our Proposed SEI Method Using SFEBLN

In this section, we introduce the proposed SEI method using SFEBLN, including its external signal processing nodes, internal signal processing nodes, feature mapping nodes, and enhancement nodes. First, in order to better understand the proposed SFEBLN in this article, we provide the preliminaries of the BLS. Then, the structure and workflow of SFEBLN are discussed. Finally, we introduce several classical signal-processing algorithms can be used within SFEBLN.

A. Preliminaries

Almost all data-driven AI models proposed in recent years can be described as in Fig. 3. For SEI problems, the RF signals

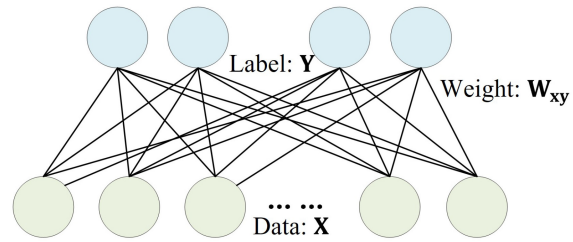


Fig. 3. Illustration of the basic structure of BLS.

are 1-D I/Q samples. Hence, the data matrix $\{\mathbf{X}|\mathbf{X} \in \mathbb{R}^{n\times l}\}$ for SEI can be defined as follows:

$$\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1l} \\ x_{21} & x_{22} & \cdots & x_{2l} \\ & & \vdots & \\ x_{n1} & x_{n2} & \cdots & x_{nl} \end{bmatrix} \tag{6}$$

where n and l are the number of data samples and the length of each sample, respectively. The real labels of the corresponding devices $\{\mathbf{Y}|\mathbf{Y} \in \mathbb{R}^{n\times c}\}$ can be similarly defined as follows:

$$\mathbf{Y} = \big[\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\big]^T = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ & & & \vdots & \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \tag{7}$$

where n and c represent the number of samples and the number of the corresponding emitter categories. Then, the simplest SEI problem can be expressed as follows:

$$\mathbf{Y} = \mathbf{X}\mathbf{W}_{xy}. \tag{8}$$

For known data and the corresponding labels, the weight $\mathbf{W}_{xy}$ can be solved by the inversion operation as follows:

$$\mathbf{W}_{xy} = \mathbf{X}^{-1}\mathbf{Y}. \tag{9}$$

Finally, $\mathbf{W}_{xy}$ shall be applicable to the identification problem under the same data distribution. That is, if the RF signals to be identified $\mathbf{X}_{\mathrm{test}}$ and the known data $\mathbf{X}$ come from the same specific domain $\mathcal{D} = \{\mathcal{X}, P_{\mathcal{X}}\}$, the corresponding labels of $\mathbf{X}_{\mathrm{test}}$ can be identified as follows:

$$\mathbf{Y}_{\mathrm{test}} = \mathbf{X}_{\mathrm{test}}\mathbf{W}_{xy} \tag{10}$$

where $\mathcal{X}(\mathbf{X}_* \in \mathcal{X})$ is the sample domain and $P_{\mathcal{X}}$ is the marginal probability distribution of the sample domain.

However, there still exist two serious flaws making this simple solution ineffective.

1) The matrix $\mathbf{X}$ could be huge, and it will be very difficult to find the corresponding inverse operation, and maybe $\mathbf{X}^{-1}$ does not exist.
2) The lack of nonlinear operations makes it unable to fit complex problems.

For the first problem, we usually solve for $\mathbf{X}^{-1}$ by pseudo-inverse or ridge regression. The pseudo-inverse operation is written as follows:

$$\widehat{\mathbf{W}}_{xy} = \arg\min_{\mathbf{W}_{xy}} \big\|\mathbf{X}\mathbf{W}_{xy} - \mathbf{Y}\big\|_2^2. \tag{11}$$

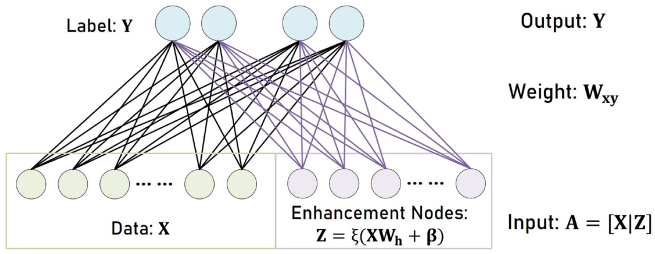


Fig. 4. Illustration of the RVFLNN.

If $\mathbf{X}$ is full column rank, then $\mathbf{X}^{\dagger} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, and the optimal weight is

$$\widehat{\mathbf{W}}_{xy} = \mathbf{X}^{\dagger}\mathbf{Y} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y} \tag{12}$$

where the pseudo-inverse $\widehat{\mathbf{W}}_{xy}$ can be regarded as the optional weight. The ridge regression is a supplement to the least square regression, which has lost the unbiased property in exchange for high numerical stability, and thus obtaining high calculation accuracy. The ridge regression methods and the approximate solution of pseudo-inverse are expressed as follows:

$$\widehat{\mathbf{W}}_{xy} = \arg\min_{\mathbf{W}_{xy}} \left\|\mathbf{XW}_{xy} - \mathbf{Y}\right\|_2^2 + \lambda\left\|\mathbf{X}_{xy}\right\|_2^2 \tag{13}$$

$$\mathbf{X}^{\dagger} = \lim_{\lambda\to 0}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T \tag{14}$$

and then the weight can be solved as $\widehat{\mathbf{W}}_{xy} = \mathbf{X}^{\dagger}\mathbf{Y}$.

Furthermore, the second problem can be solved as in Fig. 4. It is obvious that linear operations cannot fit all mathematical models. Pao et al. [53], [54] proposed a random vector functional-link neural network (RVFLNN) to solve this problem. The authors proved the universal approximation theorem in [55], which provides a theoretical basis for the BLS and the SFEBLN proposed in this article. The enhancement nodes are obtained by a nonlinear transformation of input data $\mathbf{X}$, which is defined as follows:

$$\mathbf{Z} = \xi(\mathbf{XW}_h + \boldsymbol{\beta}) \tag{15}$$

where $\xi(\cdot)$ represents the nonlinear transformation function, $\mathbf{W}_h$ and $\boldsymbol{\beta}$ are the transformation weight and bias for function $\xi(\cdot)$, respectively. Then, the input layer of RVFLNN is composed of data $\mathbf{X}$ and enhancement nodes $\mathbf{Z}$, which is expressed as $\mathbf{A} = [\mathbf{X}|\mathbf{Z}]$. Finally, the goal of RVFLNN is to obtain the network weight $\mathbf{W}_{xy}$ through pseudo-inverse or ridge regression. Theoretically, as long as there are enough enhancement nodes, the network can have enough nonlinear ability to approximate any function.

B. Proposed SFEBLN Algorithm

Since both DL and BLS are designed for CV problems, here we propose SFEBLN to handle RF signals for SEI tasks. The structure of the proposed SFEBLN method is shown in Fig. 5. It is composed of two signal processing nodes (external and internal), mapped feature nodes, and enhancement nodes. The key idea of SFEBLN is the external and internal signal processing nodes, which are used to extract the interpretable features of input signals. The four parts of SFEBLN are introduced as follows.

1) External Signal Processing Module: The external signal processing module is used to preprocess the original input RF signals, aiming at data cleaning, smoothing, augmentation, and so on, which can be defined as follows:

$$\tilde{\mathbf{X}}_{ex} = f_{ex}(\mathbf{X}, \boldsymbol{\Lambda}) \tag{16}$$

where $\tilde{\mathbf{X}}_{ex}$ is the output of external signal processing nodes, $\mathbf{X}$ and $\boldsymbol{\Lambda}$ are the input data and operator may be used in $f_{ex}(\cdot)$. It should be noted that the external signal processing nodes apply to each sample $\boldsymbol{x}_i$ of input data $\mathbf{X}$. Here, three classical external signal processing methods for a signal sequence with l length are expressed as follows:

$$\tilde{\boldsymbol{x}}_{ex}^i(k) = \sum_{n=0}^{l-1}\boldsymbol{x}^i(n)\lambda(k-n), 0 \le n, k \le l-1 \tag{17}$$

$$\tilde{\boldsymbol{x}}_{ex}^i(k) = \lambda\left[\boldsymbol{x}^i(n), win\right], 0 \le n, k \le l-1 \tag{18}$$

$$\tilde{\boldsymbol{x}}_{ex}^i = (x_{l-k+1}, \ldots, x_l, |x_1, \ldots, x_{l-k}), 0 \le k \le l-1. \tag{19}$$

The above (17)–(19) show signal convolution, windowed pooling, and signal shifting methods, respectively. Where $\lambda(\cdot)$ represents the convolution function in (17) and the max or average function in (18).

2) Mapped Feature Nodes: The mapped feature nodes $\mathbf{Z}$ are linear transformation of the input data $\mathbf{X}$, which is expressed as follows:

$$\mathbf{Z} = [\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_m] \tag{20}$$

where $\mathbf{Z}$ is composed of m mapped feature windows. $\mathbf{Z}_i$ ($i = 1, 2, \ldots, m$) represents the ith mapped feature windows, which is defined as follows:

$$\mathbf{Z}_i = \zeta\left(\tilde{\mathbf{X}}_{ex}\mathbf{W}_{z_i} + \boldsymbol{\beta}_{z_i}\right) \tag{21}$$

where $\zeta(\cdot)$ is the activation function for linear transformation, $\tilde{\mathbf{X}}_{ex}$ is the output of the external signal processing nodes, and $\mathbf{W}_{z_i}$ and $\boldsymbol{\beta}_{z_i}$ are the corresponding weight and bias of function $\zeta(\cdot)$, respectively. Here, two adjustable hyper-parameters include the length of the mapped feature window $\widetilde{m}$ and the number of mapped feature windows m. The parameter $\widetilde{m}$ will affect the shape of $\mathbf{W}_{z_i} \in \mathbb{R}^{l\times\widetilde{m}}$. And, the output shape of $\mathbf{Z}_i$ is defined as $\{\mathbf{Z}_i \in \mathbb{R}^{n\times\widetilde{m}} | i = 1, 2, \ldots, m\}$. The other parameter m determines the shape of the mapped feature nodes $\mathbf{Z} \in \mathbb{R}^{n\times(\widetilde{m}m)}$. For the broad learning network, both the weight $\mathbf{W}_{z_i}$ and the bias $\boldsymbol{\beta}_{z_i}$ are randomly initialized. In addition, the broad learning network is a one-layer forward network, which also leads to randomness in the results.

3) Enhancement Nodes: The enhancement nodes are used to further improve the fitting ability of the network. As mentioned before, theoretically, as long as there are unlimited nonlinear transformation nodes, the single-layer feedforward network can fit any problem. The enhancement nodes are designed to further broaden the network, which can be expressed as follows:

$$\mathbf{H} = \left[\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_p\right] \tag{22}$$

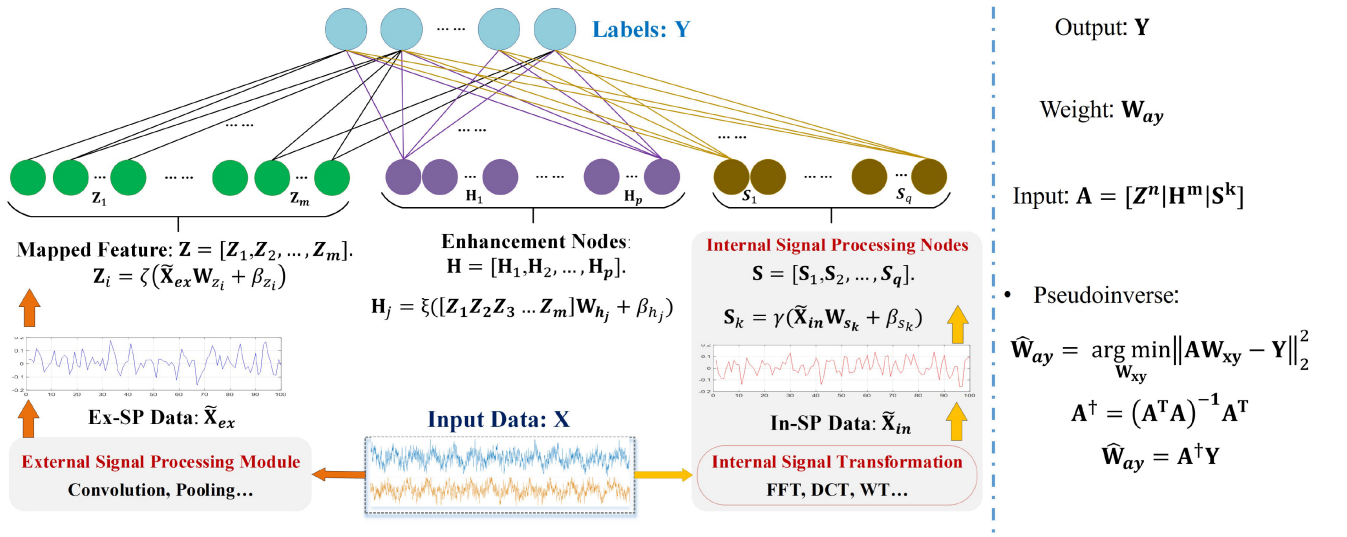


Fig. 5. Illustration of the proposed SFEBLN method for SEI applications.

where $\mathbf{H}_j (j = 1, 2, \ldots, p)$ denotes the jth enhancement nodes window, which is defined as follows:

$$\begin{aligned}\mathbf{H}_j &= \xi\left(\mathbf{Z}\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j}\right)\\ &= \xi\left([\mathbf{Z}_1\mathbf{Z}_2, \ldots \mathbf{Z}_m]\mathbf{W}_{h_j} + \boldsymbol{\beta}_{h_j}\right)\end{aligned} \tag{23}$$

where $\xi(\cdot)$ is the activation function for the enhancement nodes' transformation function, $\mathbf{Z} = [\mathbf{Z}_1\mathbf{Z}_2 \ldots \mathbf{Z}_m]$ is the output of the mapped feature nodes, and $\mathbf{W}_{h_j}$ and $\boldsymbol{\beta}_{h_j}$ are the corresponding weight and bias of function $\xi(\cdot)$, respectively. Similarly, there are two adjustable hyper-parameters that need to be determined: 1) the size of each enhance nodes window $\tilde{p}$ and 2) the number of enhance nodes windows p. The parameter $\tilde{p}$ will affect the shape of $\mathbf{W}_{h_j} \in \mathbb{R}^{(\tilde{m}m)\times\tilde{p}}$. And, the output shape of $\mathbf{H}_j$ is defined as $\{\mathbf{H}_j \in \mathbb{R}^{n\times\tilde{p}} | j = 1, 2, \ldots, p\}$. The other parameter p will determine the final output shape of the enhancement nodes $\mathbf{H} \in \mathbb{R}^{n\times(\tilde{p}p)}$.

4) Internal Signal Processing Nodes: As shown in Fig. 2, the internal signal processing nodes are generated after two steps: internal signal transformation and nonlinear transformation. The internal signal transformation usually chooses discrete signal transformation algorithms, which are expressed as follows:

$$\tilde{\mathbf{X}}_{in} = g_{in}(\mathbf{X}, \Phi) \tag{24}$$

where $\tilde{\mathbf{X}}_{in}$ is the output after internal signal transformation, $\mathbf{X}$ and Ψ are the input data and operator that may be used in $g_{in}(\cdot)$. Hence, $\tilde{\mathbf{X}}_{in}$ is considered as the transform domain signal of RF signals. As the same as external nodes, internal signal processing nodes apply to each sample $\boldsymbol{x}_i$ of input data $\mathbf{X}$. Here, we just show some classical internal signal processing methods as follows:

$$\tilde{\boldsymbol{x}}_{in}^i(k) = \sum_{n=0}^{N-1} \boldsymbol{x}^i(n) e^{-j\frac{2\pi nk}{N}}, 0 \le k \le N-1 \tag{25}$$

$$\tilde{\boldsymbol{x}}_{in}^i(k) = \begin{cases} \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} \boldsymbol{x}^i(n) \cos\frac{(2n+1)k\pi}{2N}, & k = 0 \\ \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} \boldsymbol{x}^i(n) \cos\frac{(2n+1)k\pi}{2N}, & 1 \le k \le N-1 \end{cases} \tag{26}$$

$$\tilde{\boldsymbol{x}}_{in}^i(k) = \sum_{n=0}^{N-1} \boldsymbol{x}^i(n)\phi(n-k)e^{-j2\pi fn}, 0 \le k \le N-1 \tag{27}$$

where N is the number of transformation points of finite length, and $\phi(\cdot)$ in (27) represents the windows function. The above equations (25)–(27) show discrete Fourier transform (DFT), discrete cosine transform (DCT), and STFT. However, the $\tilde{\mathbf{X}}_{in}$ cannot be the input to the network nodes before processing. We define internal signal processing nodes as follows:

$$\mathbf{S} = \left[\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_q\right] \tag{28}$$

where $\mathbf{S}$ is composed of q mapped feature windows and $\mathbf{S}_k$ $(k = 1, 2, \ldots, q)$ represents the kth internal signal processing windows, which is defined as follows:

$$\mathbf{S}_k = \gamma\left(\tilde{\mathbf{X}}_{in}\mathbf{W}_{s_k} + \boldsymbol{\beta}_{s_k}\right) \tag{29}$$

where $\gamma(\cdot)$ is the activation function for the enhancement nodes transformation function, $\mathbf{W}_{s_k}$ and $\boldsymbol{\beta}_{s_k}$ are the corresponding weight and bias of function $\gamma(\cdot)$, respectively. In this part, we need to set two hyper-parameters: the width of each internal signal processing window $\tilde{q}$ and the number of windows q. The randomly initialized weights for internal signal processing nodes follow the shape $\mathbf{W}_{s_k} \in \mathbb{R}^{N\times\tilde{q}}$. Then, internal signal processing nodes of the kth window is obtained as $\{\mathbf{S}_k \in \mathbb{R}^{n\times\tilde{q}|k=1,2,\ldots,q}\}$. And, the final internal signal processing nodes are expressed as $\mathbf{S} \in \mathbb{R}^{n\times(\tilde{q}q)}$.

5) Nonlinear Activation Functions: We briefly introduce several classical activation functions that can be chosen as $\zeta(\cdot)$, $\xi(\cdot)$, and $\gamma(\cdot)$ in the proposed SFEBLN

$$\text{out} = \tan\text{sig}(a) = \frac{2}{1+e^{-2a}} - 1 \tag{30}$$

$$\text{out} = \text{sigmoid}(a) = \frac{1}{1+e^{-a}} \tag{31}$$

$$\text{out} = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{32}$$

$$\text{out} = \text{leakyRelu}(a) = \max(0, a) + \alpha\min(0, a). \tag{33}$$

Algorithm 1: Workflow of the Proposed SFEBLN for Fast SEI

Input: RF signals for training and testing: $\mathbf{X}_{train}$, $\mathbf{X}_{test}$;
Corresponding real labels for training: $\mathbf{Y}_{train}$;
Hyper-parameters of the network: $\widetilde{m}$, m, $\widetilde{p}$, p, $\widetilde{q}$, q;
Output: Identification results: $\mathbf{Y}_{id}$;

[Training stage]:
Load $\mathbf{X}_{train} \in \mathbb{R}^{n\times l}$, $\mathbf{Y}_{train} \in \mathbb{R}^{n\times c}$;
if *external signal processing module* **then**
  Switch appropriate function from (17)∼(19);
  Compute $\widetilde{\mathbf{X}}_{ex} \in \mathbb{R}^{n\times l'}$ by (16);
end
if *mapped feature nodes* **then**
  Randomly initialize $\mathbf{W}_{z_i} \in \mathbb{R}^{l'\times\widetilde{m}}$ and $\boldsymbol{\beta}_{z_i}$;
  Compute nodes $\mathbf{Z} \in \mathbb{R}^{n\times(\widetilde{m}m)}$ by (20) and (21);
  Save all $\mathbf{W}_z$ and $\boldsymbol{\beta}_z$;
end
if *enhancement nodes* **then**
  Randomly initialize $\mathbf{W}_{h_j} \in \mathbb{R}^{(\widetilde{m}m)\times\widetilde{q}}$ and $\boldsymbol{\beta}_{h_j}$;
  Compute nodes $\mathbf{H} \in \mathbb{R}^{n\times(\widetilde{p}p)}$ by (22), (23);
  Save all $\mathbf{W}_h$ and $\boldsymbol{\beta}_h$;
end
if *internal signal processing module* **then**
  Switch appropriate function from (25)∼(27);
  Computer $\widetilde{\mathbf{X}}_{in} \in \mathbb{R}^{n\times N}$ by (24);
  Randomly initialize $\mathbf{W}_{s_k} \in \mathbb{R}^{N\times\widetilde{q}}$ and $\boldsymbol{\beta}_{s_k}$;
  Compute nodes $\mathbf{S} \in \mathbb{R}^{n\times(\widetilde{q}q)}$ by (28) and (29);
  Save all $\mathbf{W}_s$ and $\boldsymbol{\beta}_s$;
end
Concatenate all nodes as input layer $\mathbf{A} = [\mathbf{Z}|\mathbf{H}|\mathbf{S}]$;
Establish the equation to be solved $\mathbf{Y}_{train} = \mathbf{A}\mathbf{W}_{ay}$;
Obtain the solution of pseudo-inverse $\mathbf{W}_{ay}=\mathbf{A}^{\dagger}\mathbf{Y}_{train}$;
[Testing stage]:
Load $\mathbf{X}_{test}$, $\mathbf{W}_z$, $\boldsymbol{\beta}_z$, $\mathbf{W}_h$, $\boldsymbol{\beta}_h$, $\mathbf{W}_s$, $\boldsymbol{\beta}_s$, $\mathbf{W}_{ay}$;
Input $\mathbf{X}_{test}$ and compute mapped feature nodes $\mathbf{Z}_t$, enhancement nodes $\mathbf{H}_t$ and internal signal processing nodes $\mathbf{S}_t$ using the corresponding weights $\mathbf{W}_z$, $\mathbf{W}_h$, $\mathbf{W}_s$ and bias $\boldsymbol{\beta}_z$, $\boldsymbol{\beta}_h$, $\boldsymbol{\beta}_z$, respectively;
Concatenate all nodes as input layer $\mathbf{A}_t = [\mathbf{Z}_t|\mathbf{H}_t|\mathbf{S}_t]$;
Get the identification results by $\mathbf{Y}_{id} = \mathbf{A}_t\mathbf{W}_{ay}$;

The above (30)–(33) describe four commonly used nonlinear activation functions, which have been built into the proposed SFEBLN method in this study.

C. Summary and Workflow

In summary, the proposed SFEBLN has a single-layer forward propagation only network architecture. Furthermore, the SFEBLN does not need to iteratively update network weights. SFEBLN only needs to generate a large number of computing nodes and complete a pseudo-inverse operation. These characteristics make SFEBLN an efficient network structure. The most important thing is that all its operations can be executed on the CPU platform, rather than the dedicated and expensive GPU. The workflow of SFEBLN can be found in Algorithm 1.

Furthermore, the computing complexity of SFEBLN can be denoted as follows:

$$O(\text{SFEBLN}) = (X+1)\times(\tilde{m}m) + \left[(\tilde{m}m)+1\right]\times(\tilde{p}p) + \left[(\tilde{p}p)+1\right]\times(\tilde{q}q) + (\tilde{m}m+\tilde{p}p+\tilde{q}q+1)\times C \tag{34}$$

where X is the number of sampling points for each signal sample, $\tilde{m}m$, $\tilde{p}p$, and $\tilde{q}q$ are the number of mapped feature nodes, enhancement nodes, and signal processing nodes, and C is the number of identification tasks.

V. Experimental Results

This section presents our experiment study of the proposed SFEBLN and the corresponding comparison methods. We will introduce the data sets, the simulation platform, the parameter settings, and simulation results. And then we will provide our analysis and discussion of the results, including performance and overhead, robustness analysis, and stability analysis. The data sets used in this article and the simulation codes can be found at Github [56].

A. Data Set and Experiment Setup

1) Data Set: We choose the open source ADS-B data set [29] in our study. The ADS-B system is widely used to monitor the status of aircrafts, which is very important for the safety. The ADS-B signals are collected by a universal software radio peripheral (USRP) SM200B device, equipped with a 1090-MHz omnidirectional antenna. The sampling frequency is 50 MHz, center frequency and bandwidth are 1090 and 10 MHz, respectively, and the gain of signals is 30 dB. More details of the data set can be found in paper [29] and Github. Considering that the original data provided 26 613 ADS-B signal samples of 1713 different aircrafts in total, this article randomly selects $C = \{10, 20, 30, 50, 100, 200\}$ aircraft categories from the data set as six different SEI tasks. The size of each ADS-B signal sample is $\boldsymbol{x}_i \in \mathbb{C}^{6000\times1}$. In order to adapt to the input layer of SFEBLN, we reshape each ADS-B sample to $\boldsymbol{x}_i \in \mathbb{R}^{12000\times1}$. Then, we define the input data set of SFEBLN as $\{\mathbf{X}|\mathbf{X} \in \mathbb{R}^{n\times12000}\}$, where n represents the total number of ADS-B signal samples of c categories of aircrafts. And, the corresponding label matrix is $\{\mathbf{Y}|\mathbf{Y} \in \mathbb{R}^{n\times c}\}$. In addition, the ratio of training, validation, and testing data is set as 6:2:2 for each method.

2) Simulation Platform: All the simulations and experiments are carried out on a workstation with CentOS 7.0. The workstation is equipped with two Intel Xeon Silver 4210R CPUs and four Nvidia RTX 2080Ti GPUs. It also has 256-GB random access memory (RAM).

3) Hyper-Parameter Setting: As mentioned above, there are six critical hyper-parameters need to be set manually (excluding the selections of operator function and activation function). Through many attempts, we directly give the optimal parameter selection here. It should be noted that this does not mean that we provide an optimal solution of the entire problem, but the subsequent experimental results are presented in part from these parameter settings.

1) *Mapped Feature Nodes:* $\widetilde{m} = 10$ and $m = 15$.
2) *Enhancement Nodes:* $\widetilde{p} = 300$ and $p = 10$.
3) *Internal Signal Processing Nodes:* $\widetilde{q} = 100$ and $q = 1$.

Meanwhile, we choose the average pooling function (18) as external signal processing nodes and the DFT function (25) with $N = 2048$ as the internal signal transformation.

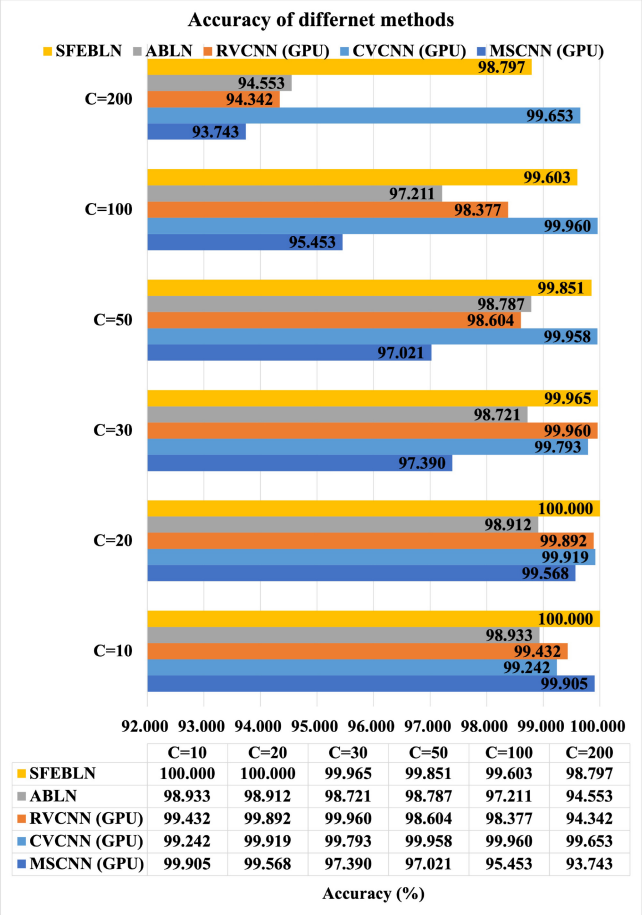
4) State-of-the-Art SEI Methods: To validate the performance and advantages of the proposed SFEBLN method, we carry out extensive comparative experiments with state-of-the-art SEI algorithms. The selected SEI solutions include one broad learning-based method, three DL-based methods, and two ML-based methods.

1) *ABLN [51]:* This is a BLS-based method called ADBLN. Actually, we choose it as our benchmark in all the experiments.
2) *RVCNN [58]:* This is a DL-based algorithm called real-valued CNN (RVCNN). It is composed of nine convolutional layers and three fully connected layers. The detailed hyper-parameters of RVCNN can be found in the original paper.
3) *CVCNN [58]:* This is the complexed-value version of RVCNN, called complexed-valued CNN (CVCNN). The structure of both CVCNN and RVCNN are the same, but CVCNN only utilizes the complex convolutional layers.
4) *MSCNN [57]:* This is a DL-based method, which is composed of three branches of convolutional layers with different kernel size. As in RVCNN/CVCNN, three fully connected layers are used as classifier for MSCNN.
5) *RForest:* This is a traditional ML-based method called random forest (RForest). It is a commonly used classifier in ML, with fast computing speed and high efficiency.
6) *SVM:* This is a traditional ML-based method called SVM. It is also a common classifier in ML.

B. Advantages of Performance and Overhead

In this part, we will show the advantages of identification performance and computation overhead of the proposed SFEBLN. We select six different tasks with $C = \{10, 20, 30, 50, 100, 200\}$, and choose three DL-based methods, MSCNN, RVCNN, and CVCNN, in this experiments. Considering that SFEBLN and ABLN just need CPU as a computing platform, we run MSCNN and RVCNN with both CPU and GPU for a better contrast effect. First, we compare the identification accuracy of the four methods. Then, we compare the computing overhead of the four algorithms. For the DL-based methods, we provide the overhead based on both GPU and CPU.

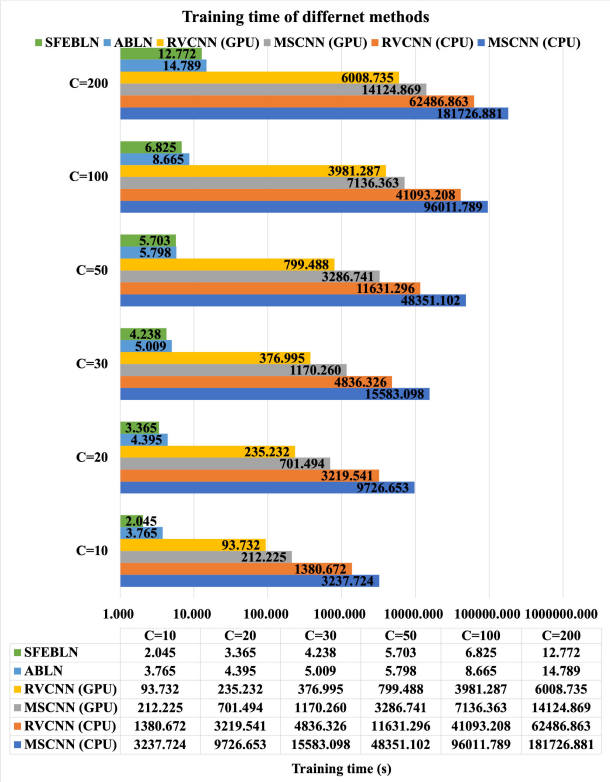
As shown in Fig. 6, the proposed SFEBLN is able to achieve 100% identification accuracy in task $C = \{10, 20\}$, more than 99% accuracy in tasks $C = \{30, 50, 100\}$, and more than 98% accuracy in the most difficult task $C = \{200\}$. From the figure, we find that the results of ABLN, RVCNN, and MSCNN are worse than SFEBLN in all the tasks. However, the CVCNN shows a better identification accuracy than SFEBLN in tasks $C = \{50, 100, 200\}$. CVCNN is an improved version of RVCNN, which not only improves its performance, but also leads to a huge computing overhead.



	C=10	C=20	C=30	C=50	C=100	C=200
SFEBLN	100.000	100.000	99.965	99.851	99.603	98.797
ABLN	98.933	98.912	98.721	98.787	97.211	94.553
RVCNN (GPU)	99.432	99.892	99.960	98.604	98.377	94.342
CVCNN (GPU)	99.242	99.919	99.793	99.958	99.960	99.653
MSCNN (GPU)	99.905	99.568	97.390	97.021	95.453	93.743

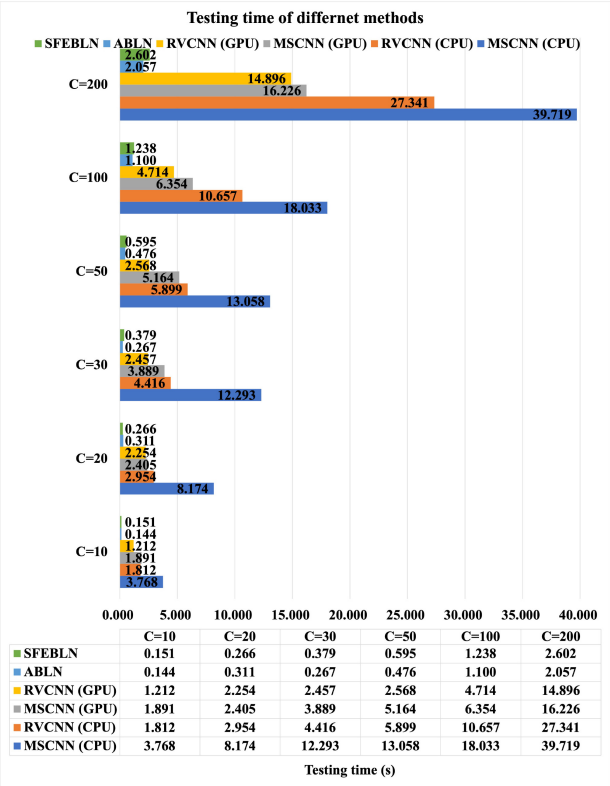
Fig. 6. Illustration of the identification accuracy of different SEI methods. SFEBLN versus ABLN versus RVCNN versus MSCNN.

In addition to the SEI performance, we should also pay attention to the computing efficiency. Therefore, this article provides a comparison of the corresponding training time and testing time, both in seconds. It should be noted that all the comparison experiments in this article use the same computing platform (in Section V-A2). And for a better comparison, we have conducted the training and testing using both GPU and CPU for the DL methods. The computation overhead of CVCNN is almost four times that of RVCNN, and it runs very slowly on the CPU. This part does not compare CVCNN, and we can refer to the results of RVCNN to calculate CVCNN. The computing overhead experimental results are shown in Fig. 7, where Fig. 7(a) and (b) show the training and testing times of different methods, respectively. It should be noted that the training times of different methods have exponential difference, the time axis in Fig. 7(a) uses the logarithmic coordinate. It is not difficult to find from Fig. 7 that SFEBLN exhibits a huge time advantage in all the SEI tasks. Except the task $C = 200$, the training time of SFEBLN in any other tasks is less than 10 s, slightly improved over ABLN. Meanwhile, with the increase of task number C, SFEBLN shows more and more advantages in training time. In the task $C = 10$, the training time of SFEBLN is only 2.18% of RVCNN (GPU), 0.96% of MSCNN (GPU), 0.148% of RVCNN (CPU), and 0.063% of MSCNN (CPU). In the task



	C=10	C=20	C=30	C=50	C=100	C=200
SFEBLN	2.045	3.365	4.238	5.703	6.825	12.772
ABLN	3.765	4.395	5.009	5.798	8.665	14.789
RVCNN (GPU)	93.732	235.232	376.995	799.488	3981.287	6008.735
MSCNN (GPU)	212.225	701.494	1170.260	3286.741	7136.363	14124.869
RVCNN (CPU)	1380.672	3219.541	4836.326	11631.296	41093.208	62486.863
MSCNN (CPU)	3237.724	9726.653	15583.098	48351.102	96011.789	181726.881

(a)



	C=10	C=20	C=30	C=50	C=100	C=200
SFEBLN	0.151	0.266	0.379	0.595	1.238	2.602
ABLN	0.144	0.311	0.267	0.476	1.100	2.057
RVCNN (GPU)	1.212	2.254	2.457	2.568	4.714	14.896
MSCNN (GPU)	1.891	2.405	3.889	5.164	6.354	16.226
RVCNN (CPU)	1.812	2.954	4.416	5.899	10.657	27.341
MSCNN (CPU)	3.768	8.174	12.293	13.058	18.033	39.719

(b)

Fig. 7. Illustration of the computation overhead of different SEI methods: SFEBLN versus ABLN versus RVCNN (GPU) versus MSCNN (GPU) versus RVCNN (CPU) versus MSCNN (CPU). (a) Training time. (b) Testing time.

$C = 200$, the training time of SFEBLN is 2.13‰ of RVCNN (GPU), 0.904‰ of MSCNN (GPU), 0.204‰ of RVCNN (CPU), and 0.071‰ of MSCNN (CPU). As for testing time, SFEBLN also shows its advantages. Because the proportion of data used for testing is small, we analyze the advantages of SFEBLN from the perspective of mathematical statistics. The testing time of SFEBLN is about 17.75% of RVCNN (GPU), 12.64% of MSCNN (GPU), 9.52% of RVCNN (CPU), and 4.72% of MSCNN (CPU). We can conclude that SFEBLN has excellent computing advantages, especially in the CPU platform. The advantages of SFEBLN in computing overhead and its independence from the GPU platform provide a promising solution for online learning.

C. Robustness Analysis

In this part, we will introduce additive white Gaussian noise (AWGN) to simulate SEI tasks in imperfect environments. The model robustness is tested at different signal-to-noise ratios $SNR = \{-10, -5, 0, 5, 10, 15\}$ dB. The SNR and AWGN are defined as follows:

$$\mathrm{SNR} = 10 \cdot \log\left(\frac{P_x}{P_n}\right) \ (\mathrm{dB}) \tag{35}$$

$$\mathrm{noise} \sim N(0, P_n) = \frac{1}{\sqrt{2\pi P_n}} \cdot \exp\left(-\frac{x^2}{2P_n}\right) \tag{36}$$

where P_x and P_n are the respective power of ADS-B signals and *noise*, $N(0, P_n)$ represents a normal distribution with mean $= 0$ and variance $= P_n$. The comparison methods we selected in this part are ABLN, RVCNN, CVCNN, MSCNN, RForest, and SVM. All DL-based methods in this part are executed on GPU.

As shown in Fig. 8, the experimental results compare the robustness of SEI methods under different tasks and noises. The red solid line in the figures represents the performance of the proposed SFEBLN under different SNRs, and the rest of the dotted lines are the performance curves of the baseline SEI methods. The small window in the figure shows the enlarged local results of SNR $= \{5, 10, 15\}$ dB. The x-axis represents the SNR range $[-10, 15]$ dB, with an interval of 5 dB. The y-axis is the identification accuracy with testing data of different task C. It is easy to see from the figure that the performance of SFEBLN remains superior under different noise conditions. Only for the $C = 200$ task, when SNR $= -10$ dB, the performance of SFEBLN is lower than that of CVCNN. When SNR $= \{-10, -5, 0\}$ dB, the classical ML-based algorithms RForest and SVM are completely ineffective, while the algorithms based on DL and broad learning are still applicable, and the SFEBLN proposed in this article still maintains its advantages in worse environments. In extremely harsh environments, such as SNR $= -10$ dB, the proposed SFEBLN shows a huge decline in identification accuracy, but it is still better than the benchmark algorithm ABLN. However, the CPU-based algorithms, RForest and SVM, show complete failure under extremely harsh conditions. There is no denying that in harsh communication environments, and on the devices without GPU support, the broad learning-based methods will be the best choice. Furthermore, the proposed SFEBLN achieves a superior performance and high robustness. In different communication environments, SFEBLN can provide fast and accurate identification for SEI tasks.

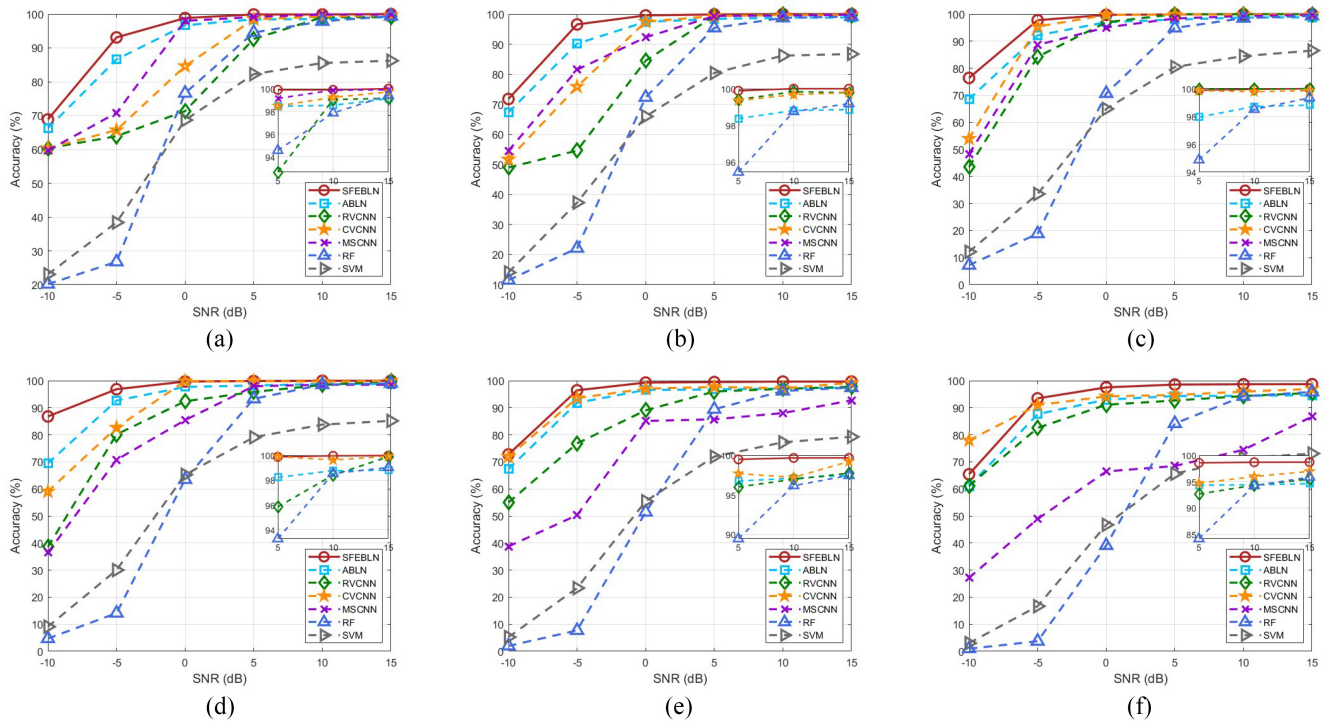


Fig. 8. Identification performance of different SEI methods with six identification tasks. SFEBLN versus ABLN versus RVCNN versus CVCNN versus MSCNN versus RForest versus SVM. (a) $C = 10$. (b) $C = 20$. (c) $C = 30$. (d) $C = 50$. (e) $C = 100$. (f) $C = 200$.

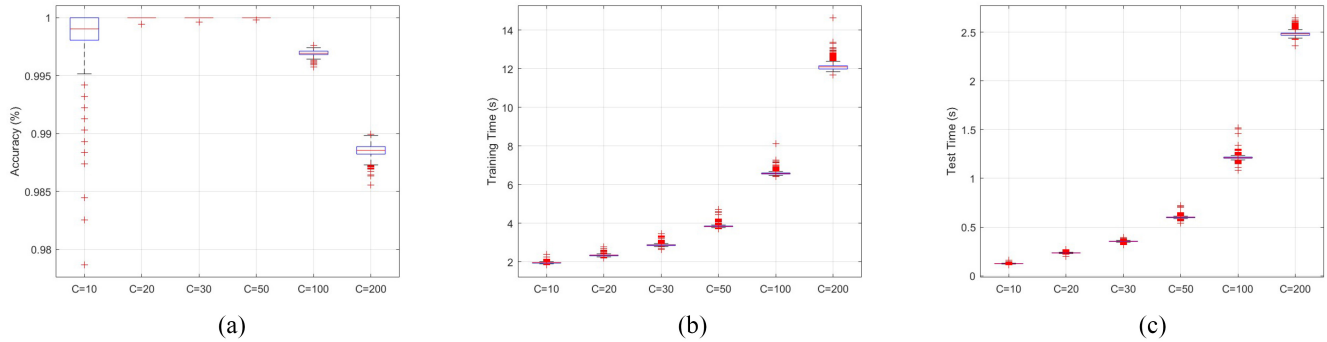


Fig. 9. Illustration of the box plots of identification accuracy, training time, and testing time for SFEBLN. (a) Accuracy versus class. (b) Training time versus class. (c) Testing time versus class.

D. Stability Analysis

In the previous analysis, we mentioned that the core mechanism of SFEBLN is to generate calculation nodes through nonlinear transformation, then find the single-layer weights, and finally obtain a feasible SEI method. All nonlinear transform nodes are generated by randomly initialized weight and offset calculation, which leads to certain randomness in SFEBLN's performance. At the same time, CPU-based computing may be affected by temperature, dominant frequency, multithreading, and other factors, which will lead to instability in SFEBLN's training time and testing time. Therefore, we conduct a Monte Carlo simulation based on six different SEI tasks $C = \{10, 20, 30, 50, 100, 200\}$, and each task is executed for 1000 times. We analyze and discuss the stability of SFEBLN through such Monte Carlo simulation results. Here, the detailed results of the 1000 Monte Carlo simulations are shown by the box plots in Fig. 9. The red "+" represents the outlier, the value of the red horizontal line is the median, the two black horizontal lines, respectively, mean the maximum and minimum values in the case of removing outliers, and the upper horizontal line of the blue box is the upper quartile, while the lower one represents the lower quartile. The x-axis in the figure represents different SEI tasks C. Fig. 9(a)–(c) show the accuracy, training time, and test time of Monte Carlo simulation, respectively.

From the box plot of accuracy, when $C = 10$, there are many discrete points in the results, showing strong randomness. This problem does not occur in $C = \{20, 30, 50\}$. When $C = \{100, 200\}$, the identification performance becomes unstable again, but it is still more stable compared with $C = 10$. As for the box plot of training time and testing time, the time consumed increases with the increase of C. We observe that

when $C = \{100, 200\}$, there are more outliers in the graph, showing stronger randomness. This phenomenon shows the randomness of SFEBLN, which is caused by randomly generated weights and offsets. However, through 1000 Monte Carlo simulations, we found that the presence of randomness does not affect the stability of SFEBLN, because most experimental results are stable within an acceptable range.

VI. Conclusion

In this article, we proposed a fast and efficient SEI method called SFEBLN, which is suitable for the CPU platform. The proposed SFEBLN method is a single-layer forward propagation network, which mainly relies on nonlinear transformation of feature mapping nodes, enhancement nodes, and signal processing nodes to solve complex classification problems. Different from DL, the proposed SFEBLN method does not need to adjust the network weight through multiple rounds of back-propagation, thus it can reduce the computation overhead. Different from traditional broad learning, our proposed SFEBLN method introduces external and internal signal processing nodes to make it suitable for RF signal data, and improves the identification accuracy through signal processing algorithms. From our experimental results, we concluded that the proposed SFEBLN method achieved an excellent performance in different SEI tasks, and had robustness advantage in a poor electromagnetic environment. Compared with the state-of-the-art, SFEBLN achieved an exponential efficiency improvement in computing overhead, and it got rid of the GPU dependency. Finally, we verified the stability of the SFEBLN performance through Monte Carlo simulation experiments, addressing the concern of system instability caused by randomly initialized weights. Our research have been proved to be fast, efficient, and easy to deploy. It is able to be deployed on SEI platforms with limited computing source.

References

[1] J. M. Hamamreh, H. M. Furqan, and H. Arslan, "Classifications and applications of physical layer security techniques for confidentiality: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1773–1828, 2nd Quart., 2019.

[2] Y. Lu and L. D. Xu, "Internet of Things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2103–2115, Apr. 2019.

[3] J. Zhang, S. Rajendran, Z. Sun, R. Woods, and L. Hanzo, "Physical layer security for the Internet of Things: Authentication and key generation," *IEEE Wireless Commun. Mag.*, vol. 26, no. 5, pp. 92–98, Oct. 2019.

[4] H. Zhao, Y. Zhang, X. Huang, Y. Xiang, and C. Su, "A physical-layer key generation approach based on received signal strength in smart homes," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 4917–4927, Apr. 2022.

[5] X. Luo, Y. Liu, H.-H. Chen, and Q. Guo, "Physical layer security in intelligently connected vehicle networks," *IEEE Netw.*, vol. 34, no. 5, pp. 232–239, Oct. 2020.

[6] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5790–5798, Aug. 2021.

[7] Y. Chen, W. Hu, M. Alam, and T. Wu, "FIDEN: Intelligent fingerprint learning for attacker identification in the Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 882–890, Feb. 2021.

[8] P. Scanlon, I. O. Kennedy, and Y. Liu, "Feature extraction approaches to RF fingerprinting for device identification in femtocells," *Bell Labs Tech. J.*, vol. 15, no. 3, pp. 141–151, Dec. 2010.

[9] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar, "A review of radio frequency fingerprinting techniques," *IEEE J. Radio Freq. Identif.*, vol. 4, no. 3, pp. 222–233, Sep. 2020.

[10] W. Su, "Feature space analysis of modulation classification using very high-order statistics," *IEEE Commun. Lett.*, vol. 17, no. 9, pp. 1688–1691, Sep. 2013.

[11] Y. Wang, J. Wang, W. Zhang, J. Yang, and G. Gui, "Deep learning-based cooperative automatic modulation classification method for MIMO systems" *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4575–4579, Apr. 20202.

[12] Y. Wang, L. Guo, Y. Zhao, J. Yang, B. Adebisi, H. Gacanin, and G. Gui, "Distributed learning for automatic modulation classification in edge devices," *IEEE Wireless Commun. Lett.*, vol. 9, no. 12, pp. 2177–2181, Dec. 2020.

[13] Y. Tu, Y. Lin, C. Hou, and S. Mao, "Complex-valued networks for automatic modulation classification," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10085–10089, Sep. 2020.

[14] Y. Yang, F. Gao, C. Xing, J. An, and A. Alkhateeb, "Deep multimodal learning: Merging sensory data for massive MIMO channel prediction," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1885–1898, Jul. 2021.

[15] J. Guo, C.-K. Wen, and S. Jin, "Deep learning-based CSI feedback for beamforming in single- and multi-cell massive MIMO systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 1872–1884, Jul. 2021.

[16] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast beamforming design via deep learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1065–1069, Jan. 2020.

[17] R. Zhao et al., "A novel intrusion detection method based on lightweight neural network for Internet of Things" *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9960–9972, Jun. 2022.

[18] J. Ning et al., "Malware traffic classification using domain adaptation and ladder network for secure Industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17058–17069, Sep. 2022.

[19] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, and D. Zhao, "Flight delay prediction based on aviation big data and machine learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 140–150, Jan. 2020.

[20] Z. Gao, Y. Gao, S. Wang, D. Li, and Y. Xu, "CRISLoc: Reconstructable CSI fingerprinting for indoor smartphone localization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3422–3437, Mar. 2021.

[21] C. Chen and Z. Liu, "Broad Learning System: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[22] C. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[23] S. Feng and C. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020.

[24] X. Gong, T. Zhang, C. Chen, and Z. Liu, "Research review for broad learning system: Algorithms, theory, and applications," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 8922–8950, Sep. 2022.

[25] H. Zhao, J. Zheng, W. Deng, and Y. Song, "Semi-supervised broad learning system based on manifold regularization and broad network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 3, pp. 983–994, Mar. 2020.

[26] X. Yuan, J. Chen, N. Zhang, X. Fang, and D. Liu, "A federated bidirectional connection broad learning scheme for secure data sharing in Internet of Vehicles," *China Commun.*, vol. 18, no. 7, pp. 117–133, Jul. 2021.

[27] X. Wang, Y. Zhu, S. Han, L. Yang, H. Gu, and F. Y. Wang, "Fast and progressive misbehavior detection in Internet of Vehicles based on broad learning and incremental learning systems," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4788–4798, Mar. 2022.

[28] S. Han, K. Zhu, M. Zhou, and X. Liu, "Evolutionary weighted broad learning and its application to fault diagnosis in self-organizing cellular networks," *IEEE Trans. Cybern.*, early access, Feb. 3, 2022, doi: 10.1109/TCYB.2021.3126711.

[29] Y. Tu et al., "Large-scale real-world radio signal recognition with deep learning," *Chin. J. Aeronaut.*, vol. 35, no. 9, pp. 35–48, Sep. 2022.

[30] J. Bian et al., "Machine learning in real-time Internet of Things (IoT) systems: A survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8364–8386, Jun. 2022.

[31] A. Bhuiyan, D. Liu, A. Khan, A. Saifullah, N. Guan, and Z. Guo, "Energy-efficient parallel real-time scheduling on clustered multi-core," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2097–2111, Sep. 2020.

[32] C. Bertoncini, K. Rudd, B. Nousain, and M. Hinders, "Wavelet fingerprinting of radio-frequency identification (RFID) tags," *IEEE Trans. Ind. Electron.*, vol. 59, no. 12, pp. 4843–4850, Dec. 2012.

[33] G. Huang, Y. Yuan, X. Wang, and Z. Huang, "Specific emitter identification based on nonlinear dynamical characteristics," *Can. J. Elect. Comput. Eng.*, vol. 39, no. 1, pp. 34–41, Jan. 2016.

[34] J. Zhang, F. Wang, O. A. Dobre, and Z. Zhong, "Specific emitter identification via Hilbert–Huang transform in single-hop and relaying scenarios," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1192–1205, Jun. 2016.

[35] U. Satija, N. Trivedi, G. Biswal, and B. Ramkumar, "Specific emitter identification based on variational mode decomposition and spectral features in single hop and relaying scenarios," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 581–591, Mar. 2019.

[36] G. Gok, Y. K. Alp, and O. Arikan, "A new method for specific emitter identification with results on real radar measurements," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3335–3346, 2020.

[37] 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, Feb. 2018.

[38] P. Yin, L. Peng, J. Zhang, M. Liu, H. Fu, and A. Hu, "LTE device identification based on RF fingerprint with multi-channel convolutional neural network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Feb. 2021, pp. 1–6.

[39] M. Ramasubramanian, C. Banerjee, D. Roy, E. Pasiliao, and T. Mukherjee, "Exploiting spatio-temporal properties of I/Q signal data using 3D convolution for RF transmitter identification," *IEEE J. Radio Freq. Identif.*, vol. 5, no. 2, pp. 113–127, Jun. 2021.

[40] 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.

[41] M. Du, X. He, X. Cai, and D. Bi, "Balanced neural architecture search and its application in specific emitter identification," *IEEE Trans. Signal Process.*, vol. 69, pp. 5051–5065, 2021.

[42] Y. Lin, Y. Tu, Z. Dou, L. Chen, and S. Mao, "Contour stella image and deep learning for signal recognition in the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 34–46, Mar. 2021.

[43] Y. Peng, P. Liu, Y. Wang, G. Gui, B. Adebisi, and H. Gacanin, "Radio frequency fingerprint identification based on slice integration cooperation and heat constellation trace figure," *IEEE Wireless Commun. Lett.*, vol. 11, no. 3, pp. 543–547, Mar. 2022.

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

[45] Z. Xiao and Z. Yan, "Radar emitter identification based on novel time-frequency spectrum and convolutional neural network," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2634–2638, Aug. 2021.

[46] Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Class-incremental learning for wireless device identification in IoT," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 17227–17235, Dec. 2021.

[47] 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.

[48] Y. Yang, A. Hu, Y. Xing, J. Yu, and Z. Zhang, "A data-independent radio frequency fingerprint extraction scheme," *IEEE Wireless Commun. Lett.*, vol. 10, no. 11, pp. 2524–2527, Nov. 2021.

[49] 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, Feb. 2022.

[50] B. Chatterjee, D. Das, S. Maity, and S. Sen, "RF-PUF: Enhancing IoT security through authentication of wireless nodes using in-situ machine learning," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 388–398, Feb. 2019.

[51] Z. Xu, G. Han, L. Liu, H. Zhu, and J. Peng, "A lightweight specific emitter identification model for IIoT devices based on adaptive broad learning," *IEEE Trans. Ind. Informat.*, early access, Sep. 13, 2022, doi: 10.1109/TII.2022.3206309.

[52] Y. Li, X. Chen, Y. Lin, G. Srivastava, and S. Liu, "Wireless transmitter identification based on device imperfections," *IEEE Access*, vol. 8, pp. 59305–59314, 2020.

[53] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory,system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, May 1992.

[54] Y. H. Pao, G. H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[55] B. Igelnik and Y. H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[56] "SEI-SFEBLN." [Online]. Available: https://github.com/CodeDwan/SEI-SFEBLN

[57] Y. Zhang, Y. Peng, B. Adebisi, G. Gui, H. Gacanin, and H. Sari, "Specific emitter identification based on radio frequency fingerprint using multi-scale network," in *Proc. IEEE 96th Veh. Technol. Conf. (VTC-Fall)*, London, U.K., 2022, pp. 1–5.

[58] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, O. A. Dobre, and H. V. Poor, "An efficient specific emitter identification method based on complex-valued neural networks and network compression," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2305–2317, Aug. 2021.