IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 20, 2021, accepted December 11, 2021, date of publication December 14, 2021,
date of current version December 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3135568

Limited Data Spectrum Sensing Based on Semi-Supervised Deep Neural Network

YUPEI ZHANG[1] AND ZHIJIN ZHAO[2]
[1]School of Electronic and Information, Hangzhou Dianzi University, Hangzhou 310018, China
[2]School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Zhijin Zhao (zhaozj03@hdu.edu.cn)

This work was supported in part by the State Key Program of National Natural Science of China under Grant U19B2016, and in part by the Key Lab of Data Storage and Transmission Technology of Zhejiang Province, Hangzhou Dianzi University.

ABSTRACT Spectrum sensing methods based on deep learning require massive amounts of labeled samples. To address the scarcity of labeled samples in a real radio environment, this paper presents a spectrum sensing method based on semi-supervised deep neural network (SSDNN). Firstly, a deep neural network is established to extract the features of signals by using small amounts of labeled samples; Then, plenty of unlabeled samples are used for self-training process, and the ones with high confidence are marked with pseudo-label to expand the labeled dataset. Finally, the extended dataset is used to retrain the network. Plentiful experiments are carried out on a dataset of 124,800 samples. The results demonstrate that the proposed algorithm has good detection performance over multi-path fading channel and additive white Gaussian noise channel due to the utilization of a great deal of unlabeled dataset. When the labeled samples account for only 5% of the traditional fully supervised deep learning model and the SNR is higher than −13 dB, the detection probability of SSDNN is higher than 90%.

INDEX TERMS Cognitive radio, spectrum sensing, deep neural network, semi-supervised learning, limited data.

I. INTRODUCTION

With the rapid development of communication technology, the wireless spectrum is widely used in broadcasting, satellite, military, and other communication systems. Research shows that the utilization rate of authorized frequency band is between 15% and 80%, while the unlicensed frequency band is increasingly short [1]. As an intelligent wireless communication technology, cognitive radio (CR) [2], [3] can intelligently find available free spectrum to improve spectrum utilization. Spectrum sensing is crucial for CR [4], [5]. The secondary user (SU) detects whether there is a primary user (PU) in a specific frequency band through spectrum sensing, so as to decide if available free spectrum exists or not. Therefore, improving the accuracy of spectrum sensing can effectively enhance spectrum utilization.

Traditional spectrum sensing methods are divided into single node spectrum sensing [6]–[9] and collaborative spectrum sensing [10]–[12]. Typical single node spectrum sensing methods include energy detection [6], cyclostationary feature detection [7], matched filter detection [8], and frequency domain entropy-based methods [9]. In collaborative spectrum sensing, the fusion center makes the final decision according to the hard fusion or soft fusion rules by receiving the signal or decision result of each SU [10]. Due to the complexity of the actual communication environment, both single node spectrum sensing and collaborative spectrum sensing are required to have the ability to adapt to the complex and changeable communication environment, as well as the ability to carry out spectrum sensing quickly. However, the traditional spectrum sensing methods are not always able to meet the requirements of the real electromagnetic environment.

With the development of deep learning technology, the feature extraction ability of neural network is constantly improving, and spectrum sensing algorithms based on deep learning emerge in endlessly [13]–[16]. The spectrum sensing methods based on deep learning have strong feature extraction ability and detection performance. Deep learning makes spectrum sensing intelligent by collecting environment information and user state in the CR network for modeling and reasoning learning, so as to make it adapt to the actual

The associate editor coordinating the review of this manuscript and approving it for publication was Pietro Savazzi.

TABLE 1. Deep learning model in spectrum sensing.

Method	Learning style	Model	Input feature	PU prior information
Pan et al.[18]	supervised	CNN	cycle spectrum	OFDM frame
Xie et al.[19]	supervised	CNN	covariance matrix	-
Ozdes et al.[20]	supervised	CNN	power spectrum	-
Zheng et al. [21]	supervised	CNN	power spectrum	-
Chew et al.[22]	supervised	CNN	spectrogram	OFDM frame
Liu et al.[23]	supervised	CNN	covariance matrix	primary activity statistics
Xie et al.[24]	supervised	LSTM+CNN	covariance matrix	primary activity statistics
Subekti et al.[25]	supervised	DAE+SVM	RGB image	-

communication environment and obtain high performance [17]. Table 1 summarizes several typical spectrum sensing algorithms based on deep learning.

In [18]–[23], a convolutional neural network(CNN) is applied to spectrum sensing. Pan *et al.* proposed a spectrum sensing method for orthogonal frequency division multiplexing (OFDM) signals based on deep learning and cyclic spectrum [18]. In that paper, the cyclic autocorrelation characteristics of the OFDM signal were analyzed, and the cyclic spectrum was obtained by using the time domain smooth fast Fourier transform accumulation algorithm. Then the cyclic spectrum was converted into the gray image as the input feature. The improved CNN model based on LeNet-5 was used to extract deep features layer by layer. The numerical simulation results indicate that the proposed method has better sensing performance than the traditional methods under low signal-to-noise ratio (SNR). Xie *et al.* used the information of the activity pattern of PU to find available free spectrum [19], which can be divided into two phases: offline training and online identification. In the phase of offline training, the CNN trained model parameters using the covariance matrix of the sensing data in the current frame, the covariance matrix of historical sensing data and the labeled PU state data. In the phase of online identification, the trained CNN achieved real-time detection based on current and historical sensing data. The numerical simulation results show that the algorithm is better than the estimator detection and hidden Markov model detector. In [20], the CNN with 5 convolution layers, three maximum pooling layers and two fully connected layers were used for binary classification to determine whether the signal exists or not. The results show that the accuracy of the training set is about 98%, and the accuracy of the validation set is about 92%. In [21], the normalized signal power spectrum was used as the input of CNN, and eight kinds of modulation signals and noises were used to train the network. The simulation results show that the performance of the method is better than that of the traditional method based on the maximum-minimum eigenvalue ratio and the method based on frequency entropy. At the same time, it has a strong generalization ability and can detect various untrained signals. In the case of noise and interference, the signal spectrum sensed by SU was calculated in [22] and sent to the CNN detector for classification to determine whether a PU signal exists. Simulation results show that the performance of the CNN detector is better than that of the classical energy detector. In [23], Liu *et al.* used the sample covariance matrix as the input to generate test statistics. Thus, a CNN model with maximum posterior probability (MAP) as the cost function was designed. According to the Neyman-Pearson criterion, the spectrum sensing likelihood ratio test method based on CNN was derived. It is proved that the CM-CNN method is equivalent to the optimal estimator correlator detector under an independent identical distribution model. In [24], LSTM was combined with CNN to realize spectrum sensing. CNN was used to extract energy-related features from the covariance matrix generated by receiving data. Then the features of multiple sensing periods were input into LSTM to learn PU activity pattern, so as to further improve the detection probability, and the performance of CNN-LSTM was verified in the presence and absence of noise uncertainty. In [25], a spectrum sensing method combining deep auto-encoder (DAE) neural network and SVM was presented. The received signal was converted into an image and sent to DAE for feature learning. The results of DAE were input into SVM for classification, so as to determine whether the input signal was a PU or SU.

The above spectrum sensing methods using deep learning are all based on supervised learning, which can obtain excellent detection performance when the true-label dataset is sufficient. In a real radio environment, it is feasible to use the receiver to obtain vast amounts of PU signals, but labeling all the signals manually is very time-consuming. Moreover, there may be mislabeling. At present, the research of spectrum sensing for limited label samples is still in its infancy at home and abroad, but there are some semi-supervised learning methods in other fields [26], [27].

Inspired by the extensive usage of MIMO technology in spectrum sensing algorithms, Xie *et al.* proposed an unsupervised deep spectrum sensing algorithm (UDSS) [14], which established the variational auto-encoder Gaussian mixture model to complete the signal clustering identification. However, the high detection probability of this method relies on a large number of antennas and the significant correlation coefficient between the received signals of each antenna. Motivated by this, the problem of the spectrum sensing is considered in a general sense in this paper, regardless of the specific signal types and complex channel models. The application of semi-supervised learning in spectrum sensing

is investigated, as it only requires a few of labeled training samples at a small cost and can make the best use of a majority of unlabeled samples.

Very recently, semi-supervised learning methods mainly use artificial features and classifiers to realize self-learning. The typical classifiers mainly include support vector machine [28], [29], k-nearest neighbor [30] (KNN), Gaussian mixture model [31], [32] (GMM), and artificial neural network [33]–[35] (ANN), etc. These classifiers can achieve better performance in the case of only a bit of labeled samples. However, it is challenging to select the appropriate features and classifiers in practical application. Aiming to solve the above problems, this paper combines a semi-supervised learning method with a deep neural network, which can avoid selecting features and classifiers. The feature extraction and classifier can be combined to conduct the training.

To sum up, the contributions of this paper are as follows:

- The training phase of this method is based on semi-supervised classification. Therefore, only a small number of labeled samples are needed. Compared with the traditional spectrum sensing method based on deep learning, the proposed method significantly reduces the dependence on labeled samples.
- To make full use of slight labeled samples and a majority of unlabeled samples, a semi-supervised deep neural network is proposed to improve the classification accuracy by setting the confidence function and modifying the cross entropy loss function.
- Through extensive simulation and comparison, the advantages of the proposed detection method are verified compared with traditional spectrum sensing methods, such as the energy detection method (ED) and entropy-based method (Entropy). Meanwhile, it shows that the proposed method can effectively improve the performance of spectrum sensing. The detection probability is almost the same as that of the fully supervised deep learning method, making it a more practical solution to address the issue that it is difficult to acquire a sufficient amount of labeled dataset in deep learning.
- In addition, the effects of different sampling lengths, different false alarm probabilities, different channel fading scenarios, and different frequency offset scenarios on the detection performance of the SSDNN model are also explored.

II. METHOD AND IMPLEMENTATION

A. PROBLEM DESCRIPTION

Spectrum sensing problem can be effectively transformed into a binary hypothesis decision problem. There are two hypotheses: H_0 represents the absence of the primary user, and H_1 represents the presence of the primary user, then the N points of the received signal can be expressed as [36]:

$$\begin{aligned} H_0 &: x(n) = v(n) \\ H_1 &: x(n) = h(n)s(n) + v(n) \end{aligned} \tag{1}$$

where, $n = 0, 1, 2, \cdots, N-1$, $x(n)$ denotes the complex signal received by SU; $s(n)$ is the transmitted signal of PU; $v(n)$ is the additive white Gaussian noise (AWGN) subject to $N\left(0, \sigma^2\right)$, and $h(n)$ is the channel gain between PU and SU. Detection probability P_d and false alarm probability P_f are two important indexes to measure the performance of spectrum sensing:

$$\begin{aligned} P_d &= P\left(H_1 \mid H_1\right) \\ P_f &= P\left(H_1 \mid H_0\right) \end{aligned} \tag{2}$$

Assuming that the signal dataset received by SU is D, it is divided into labeled dataset $D^L = \left(\boldsymbol{X}^L, \boldsymbol{Y}^L\right) = \left\{\left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \left(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}\right), \cdots, \left(\mathbf{x}^{(l)}, \mathbf{y}^{(l)}\right)\right\}$ and unlabeled dataset $D^U = \boldsymbol{X}^U = \left\{\mathbf{x}^{(l+1)}, \mathbf{x}^{(l+2)}, \cdots, \mathbf{x}^{(l+u)}\right\}$, where $x^{(i)}$ is the ith sample, $y^{(i)}$ is the true label of $x^{(i)}$, l is the total number of labeled samples, and u is the total number of unlabeled samples, on the premise of $l \ll u$. $\mathbf{y}^{(i)} \in [0, 1]$ represents H_0 or H_1, respectively. The purpose of SSDNN is to pre-train the neural network according to the labeled dataset D^L and mark the reliable unlabeled dataset $\mathbf{X}^{U'}$ in X^U with $\mathbf{Y}^{U'}$. Then the extended dataset $D' = \left\{\left(\mathbf{X}^L, \mathbf{Y}^L\right), \left(\mathbf{X}^{U'}, \mathbf{Y}^{U'}\right)\right\}$ is used to retrain a more powerful classifier model with better classification performance.

B. PROPOSED METHOD

Figure 1 highlights the framework of SSDNN, which is mainly composed of the pre-training phase, the semi-supervised learning phase, as well as the training and sensing phase.

In the pre-training phase, DNN is pre-trained according to the labeled dataset $D^L = \left(\boldsymbol{X}^L, \boldsymbol{Y}^L\right)$; In the semi-supervised learning phase, the unlabeled dataset X^U is input, and the dataset $\mathbf{X}^{U'}$ with a high confidence level is marked with pseudo labels $\mathbf{Y}^{U'}$ by using the pre-trained model. In this phase, there are usually several iterations. In the training and sensing phase, the labeled samples and the iteratively generated pseudo-label samples are used as the new dataset $D' = \left\{\left(\mathbf{X}^L, \mathbf{Y}^L\right), \left(\mathbf{X}^{U'}, \mathbf{Y}^{U'}\right)\right\}$ to retrain the DNN model $g_{\mathrm{w,b}}$, maximize the posterior probability $P(\mathbf{Y} \mid \mathbf{X})$, and obtain the optimal model parameter $\mathbf{W}^*$ and $\mathbf{b}^*$.

$$\left(\mathbf{w}^*, \mathbf{b}^*\right) = \arg\max_{\mathbf{w},\mathbf{b}} P(\mathbf{Y} \mid \mathbf{X}; \mathbf{w}; \mathbf{b}) \tag{3}$$

So far, an optimal spectrum sensing classifier model $g^*_{\mathbf{w},\mathbf{b}}$ has been obtained, and the spectrum sensing can be realized after inputting the test dataset.

1) PRE-TRAINING

The pre-training phase is the critical step of the SSDNN model. Adding unlabeled samples won't improve the performance of the model until the pre-trained network reaches a certain accuracy. The training process includes forward propagation and backward propagation. Forward propagation refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the

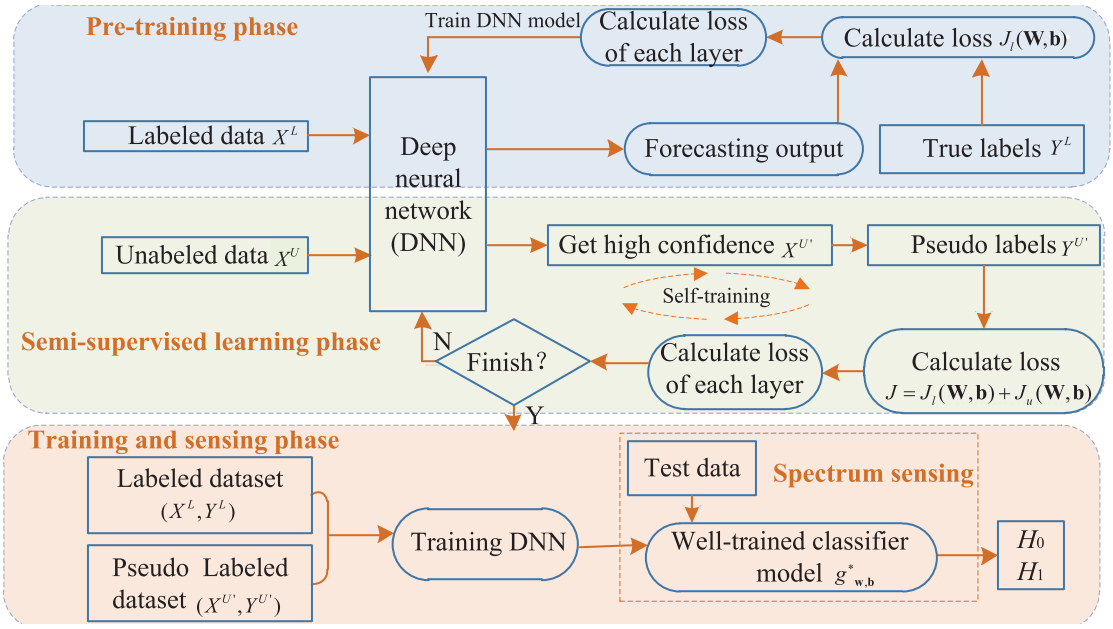


FIGURE 1. Framework of SSDNN.

input layer to the output layer according to the training set. The CNN input is set as $\mathbf{a}^1 = \mathbf{x}^{(i)}$, and the weight parameter $\mathbf{W}$ and bias parameter $\mathbf{b}$ of the network are initialized. The total number of layers of the network is K. The $k(k = 2 \sim K-1)$ layer may be one of the convolution layer, pooling layer, and fully connected layer. The Kth layer is the output $\mathbf{a}^{i,K} = \left[\text{prob}\left(\mathbf{x}^{(i)}, H_1\right), \text{prob}\left(\mathbf{x}^{(i)}, H_0\right)\right]$ of the softmax classifier, which represents the probability that the ith sample belongs to H_1 or H_0, respectively. The label corresponding to the maximum probability is selected as the classification result $\mathbf{y}^{(i)}$ of the ith sample, and $\text{prob}\left(\mathbf{x}^{(i)}, H_1\right) + \text{prob}\left(\mathbf{x}^{(i)}, H_0\right) = 1$. The output of each layer is as follows.

If the kth layer is a convolution layer, the output is:

$$\mathbf{a}^{i,k} = \sigma\left(\mathbf{z}^{i,k}\right) = \sigma\left(\mathbf{a}^{i,k-1} * \mathbf{W}^k + \mathbf{b}^k\right) \tag{4}$$

where i represents the sample number, k represents the layer number, and $\sigma(\cdot)$ represents the activation function, $\mathbf{z}^{i,k} = \mathbf{a}^{i,k-1} * \mathbf{W}^k + \mathbf{b}^k$.

If the kth layer is a pooling layer, the output is:

$$\mathbf{a}^{i,k} = \text{pool}\left(\mathbf{a}^{i,k-1}\right) \tag{5}$$

where pool$(\cdot)$ refers to the process of reducing the dimension of input features according to the size of the pooling template and pooling standard.

If the kth layer is a fully connected layer, the output is:

$$\mathbf{a}^{i,k} = \sigma\left(\mathbf{z}^{i,k}\right) = \sigma\left(\mathbf{a}^{i,k-1}\mathbf{W}^k + \mathbf{b}^k\right) \tag{6}$$

For layer K (output layer), the calculation formula is as follows:

$$\mathbf{a}^{i,K} = \text{softmax}\left(\mathbf{z}^{i,K}\right) = \text{softmax}\left(\mathbf{a}^{i,K-1}\mathbf{W}^K + \mathbf{b}^K\right) \tag{7}$$

Then the binary cross entropy loss between the expected output and the real output is calculated by using equation (8).

$$\begin{aligned} &J_l(\mathbf{W}, \mathbf{b}) \\ &= -\frac{1}{l}\sum_{i=1}^{l}\mathbf{y}^{(i)}\log\left(\text{prob}\left(\mathbf{x}^{(i)}, H_1\right)\right) \\ &\quad + \left(1-\mathbf{y}^{(i)}\right)\log\left(\text{prob}\left(\mathbf{x}^{(i)}, H_1\right)\right), \quad \mathbf{x}^{(i)} \in \mathbf{X}^L \end{aligned} \tag{8}$$

Comparing the error between the output value and the expected value of the network, when the error is greater than the expected value, the error is sent back to the network, and the errors of the fully connected layer, pooling layer, and convolution layer are obtained in turn. The weight parameters are adjusted and updated by the error gradient, and the network is trained again, which is called backward propagation. The propagation sensitivity (error) $\boldsymbol{\delta}^{i,K}$ of the output layer can be calculated from the cost function:

$$\boldsymbol{\delta}^{i,K} = \frac{\partial J_l(\mathbf{W}, \mathbf{b})}{\partial \mathbf{z}^{i,K}} = \frac{\partial J_l(\mathbf{W}, \mathbf{b})}{\partial \mathbf{a}^{i,K}} \odot \sigma'\left(\mathbf{z}^{i,K}\right) \tag{9}$$

where "$\odot$" denotes Hadamard product, and $\sigma'(\cdot)$ is the derivative of the activation function $\sigma(\cdot)$.

If the kth layer is a convolution layer, the error output is as follows:

$$\boldsymbol{\delta}^{i,k} = \delta^{i,k+1} * \text{rot}\,180\left(\mathbf{W}^{k+1}\right) \odot \sigma'\left(\mathbf{z}^{i,k}\right) \tag{10}$$

where rot $180(\cdot)$ represents that the convolution kernel rotates by 180 degrees, which can be realized by row symmetry transformation and column symmetry transformation. For each convolution kernel, the learning rate is set to η,

then $\mathbf{W}$ and $\mathbf{b}$ are updated as follows:

$$\mathbf{W}^k = \mathbf{W}^k - \eta \sum_{i=1}^{l} \delta^{i,k} * \mathbf{a}^{i,k-1} \tag{11}$$

$$\mathbf{b}^k = \mathbf{b}^k - \eta \sum_{i=1}^{l} \sum_{u,v} \left(\boldsymbol{\delta}^{i,k}\right)_{u,v} \tag{12}$$

where $\left(\boldsymbol{\delta}^{i,k}\right)_{u,v}$ denotes the submatrix of $\delta^{i,k}$.

If the kth layer is a pooling layer, the error output is as follows:

$$\boldsymbol{\delta}^{i,k} = \text{upsample}\left(\delta^{i,k+1}\right) \odot \sigma'\left(\mathbf{z}^{i,k}\right) \tag{13}$$

The unsample$(\cdot)$ function completes the logic of pooling error matrix amplification and error redistribution.

If the kth layer is a fully connected layer, the error output is as follows:

$$\boldsymbol{\delta}^{i,k} = \left(\mathbf{W}^{k+1}\right)^{\mathrm{T}} \boldsymbol{\delta}^{i,k+1} \odot \sigma'\left(\mathbf{z}^{i,k}\right) \tag{14}$$

The updated expressions of parameters $\mathbf{W}$ and $\mathbf{b}$ are as follows:

$$\mathbf{W}^k = \mathbf{W}^k - \eta \sum_{i=1}^{l} \delta^{i,k} \left(\mathbf{a}^{i,k-1}\right)^{\mathrm{T}} \tag{15}$$

$$\mathbf{b}^k = \mathbf{b}^k - \eta \sum_{i=1}^{l} \boldsymbol{\delta}^{i,k} \tag{16}$$

The training is not over until the error is equal to or less than expected.

2) SEMI-SUPERVISED LEARNING

Semi-supervised learning is a crucial technology to improve classification performance using unlabeled dataset [37]. The most frequently used semi-supervised learning algorithm is self-training [38], [39], which has become a new research direction in the field of machine learning and an essential branch of data mining. Moreover, it is gradually becoming a practical tool in many areas. Semi-supervised self-training attempts to automatically mark unlabeled samples with pseudo labels and add them to the labeled dataset of each learning cycle.

The pre-trained deep learning network is used to predict the classification category (H_0 or H_1) of unlabeled samples, and this category is regarded as the pseudo label of the unlabeled sample. As the initial classifier uses a small dataset, the classification performance may not be high, so the pseudo labels generated by the model are likely to be incorrect and may prevent the learning of new information. Once sufficient amounts of wrong labels are added to the self-training, it cannot improve the accuracy of classification, and even reduce the performance of the classifier. Therefore, this paper mainly enhances the proportion of true labels and reduces the interference of wrong labels to the training model by setting the confidence function and modifying the cross entropy loss function.

3) LABELING SAMPLES WITH HIGH CONFIDENCE

To find the samples with the highest correct probability from unlabeled samples, a confidence measure function is defined. Using the confidence value provided by the function, the sample with the highest correct probability that needs to be added to the next round of self-training can be obtained. The ith sample in DNN outputs two neurons $\mathbf{a}^{i,K} = \left[\text{prob}\left(\mathbf{x}^{(i)}, H_1\right), \text{prob}\left(\mathbf{x}^{(i)}, H_0\right)\right]$ from the softmax classifier. The confidence function is defined as follows:

$$\begin{aligned}&\text{Confidence}\left(\text{prob}\left(\mathbf{x}^{(i)}, H_1\right), \text{prob}\left(\mathbf{x}^{(i)}, H_0\right)\right)\\ &\quad = \left|\text{prob}\left(\mathbf{x}^{(i)}, H_1\right) - \text{prob}\left(\mathbf{x}^{(i)}, H_0\right)\right|\end{aligned} \tag{17}$$

The higher the confidence value p obtained by formula (17), the greater the probability of output as correct classification.

4) MODIFYING CROSS ENTROPY LOSS FUNCTION

If the pseudo label is regarded as the true label directly and the loss of unlabeled dataset is calculated by the formula (8), the model will bias toward the wrong training direction, since these pseudo labels are not all true labels. Therefore, the loss gradient brought by them to the model needs to be multiplied by a balance coefficient λ to avoid being excessively affected by the wrong information:

$$\begin{aligned}&J_u(\mathbf{W}, \mathbf{b})\\ &= -\frac{\lambda}{\left|\mathbf{X}^{U'}\right|} \sum_{i=1}^{\left|\mathbf{x}^{U'}\right|} \hat{\mathbf{y}}^{(i)} \log\left(\text{prob}\left(\mathbf{x}^{(i)}, H_1\right)\right)\\ &\quad + \left(1 - \hat{\mathbf{y}}^{(i)}\right) \log\left(\text{prob}\left(\mathbf{x}^{(i)}, H_0\right)\right), \mathbf{x}^{(i)} \in \mathbf{X}^{U'}\end{aligned} \tag{18}$$

where $\left|\mathbf{x}^{U'}\right|$ refers to the size of pseudo labeled dataset, and $\hat{\mathbf{y}}^{(i)}$ represents the corresponding pseudo label of $\mathbf{x}^{(i)}$. The modified cross entropy loss function is obtained by combining labeled samples and unlabeled samples.

$$J(\mathbf{W}, \mathbf{b}) = J_l(\mathbf{W}, \mathbf{b}) + J_u(\mathbf{W}, \mathbf{b}) \tag{19}$$

5) TRAINING AND SENSING

The DNN model $g_{\mathrm{w,b}}$ is retrained with a pretty small number of true-label samples and a mass of pseudo-label samples generated in the semi-supervised phase to maximize the posterior probability $P(\mathbf{Y} \mid \mathbf{X})$, so as to obtain the optimal spectrum sensing classifier $g^*_{w,b}$. In order to evaluate the spectrum sensing performance of the classifier, the test dataset is used.

The test dataset does not reuse the training dataset. It is assumed that there are M pairs of test samples $D^M = (\boldsymbol{X}^M, \boldsymbol{Y}^M) = \left\{\left(\mathbf{x}_{\text{test}}^{(1)}, \mathbf{y}_{\text{test}}^{(1)}\right), \left(\mathbf{x}_{\text{test}}^{(2)}, \mathbf{y}_{\text{test}}^{(2)}\right), \cdots \left(\mathbf{x}_{\text{test}}^{(M)}, \mathbf{y}_{\text{test}}^{(M)}\right)\right\}$. When there is a primary user, M pairs of test signal samples are generated. For the ith data sample, $\mathbf{x}_{\text{test}}^{(i)}$ is input to the classifier $g^*_{w,b}$ and the output $\mathbf{a}_{\text{test}}^{i,K} = \left[\text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_1\right), \text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right)\right]$ is obtained,

where prob $\left(\mathbf{x}_{\text{test}}^{(i)}, H_1\right)$ and prob $\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right)$ denotes the probability of belonging to signal or noise, respectively. In order to improve the sensing accuracy, the detection criteria shown in equation (20) are used in reference [21]:

$$\begin{aligned} H_0 &: 1 - \text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right) \leq \gamma \\ H_1 &: 1 - \text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right) > \gamma \end{aligned} \tag{20}$$

Setting the threshold γ and counting the number of test samples k_d that satisfy $1 - \text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right) > \gamma$, then the recognition accuracy is k_d/M, which is also the detection probability P_d under this threshold.

Similarly, when there is no primary user, M pairs of test signal samples are generated, and the number of test samples k_f satisfying $1 - \text{prob}\left(\mathbf{x}_{\text{test}}^{(i)}, H_0\right) \leq \gamma$ can be counted to obtain the false alarm probability $P_f = k_f/M$ under this threshold.

The SSDNN algorithm flow is as follows.

Algorithm 1 SSDNN Network Algorithm Flow

Input: true-label dataset $D^L = \left(\boldsymbol{X}^L, \boldsymbol{Y}^L\right)$, unlabeled dataset $D^U = \boldsymbol{X}^U$, maximum self-training times T, pseudo label confidence threshold p, balance coefficient λ, initial self-training times t = 0.

Output: get the optimal spectrum sensing classifier $g_{w,b}^*$.

1: **while** t <= T **do**
2: Input true-label dataset $D^L = \left(\boldsymbol{X}^L, \boldsymbol{Y}^L\right)$
3: Initialize SSDNN model network parameter $(\mathbf{W}, \mathbf{b})$
4: Training process : calculate the actual output according to equations (4)-(7), use equation (9) to calculate the forward propagation error of this iteration, reversely obtain the error of each layer according to equations (10), (13) and (14), and update the parameter $(\mathbf{W}, \mathbf{b})$ of each layer;
5: Load the network parameter $(\mathbf{W}, \mathbf{b})$ of the current iteration number, input D^U into the network, select the sample set with confidence higher than p, mark the pseudo label $\mathbf{Y}^{U'}$ and add it to the training set $\mathbf{X}^{U'}$, delete the sample set $\mathbf{Y}^{U'}$ from D^U, and replace equation (8) with the modified cross entropy loss function shown in equation (19)
6: **end while**
7: The labeled samples and the iteratively generated pseudo labeled samples $D' = \left\{\left(\mathbf{X}^L, \mathbf{Y}^L\right), \left(\mathbf{X}^{U'}, \mathbf{Y}^{U'}\right)\right\}$ are used as the new training set to retrain the DNN model $g_{\text{w,b}}$, and an optimal spectrum sensing classifier $g_{w,b}^*$ is obtained.

The structure of the deep neural network (DNN) is shown in Figure 2. Network input is an IQ dual channel matrix composed of receiving complex signals $x(n)$. $(n_1 \times n_2, \text{Conv}, n_3)$ represents the convolution layer of n_3 convolution cores with a size of $n_1 \times n_2$; BN represents batch standardization, and FC represents fully connected layer.

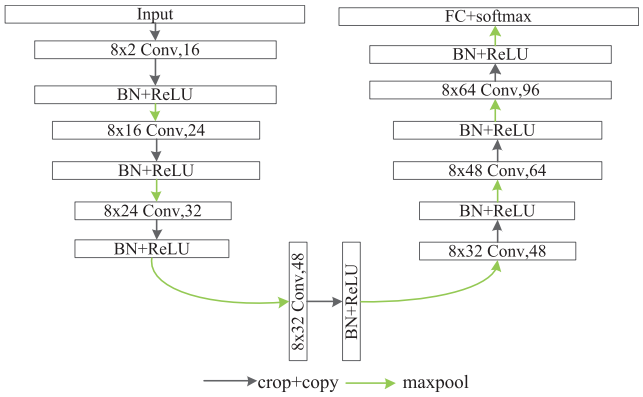


FIGURE 2. The Framework of the deep neural network.

III. EXPERIMENT AND ANALYSIS

A. DATASET

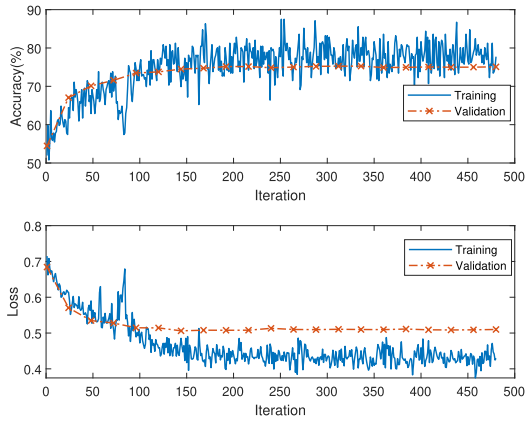
For dataset, PU signals are modulated by four types, namely, AM, BFSK, QPSK, and 16QAM, all of which suffer from Rican channel fading and are polluted by additive Gaussian white noise. 960 samples of 1024 points are generated for each modulation signal with different SNR from −20dB to 4dB with an interval of 2dB, and additive Gaussian white noise samples with the same number and points are generated with zero mean and unit variance to form a training set with a total number of 99,840 samples. Similar to the above, 1,920 samples of four modulation signals under each SNR are generated to form a test set with a total number of 24,960 samples. In total, 124,800 samples are generated to form the original dataset. Instead of directly using the received time-domain complex signal $x(n)$, it is normalized to $x(n)/\sqrt{\frac{\sum_{n=1}^{N}|x(n)|^2}{N}}$ before training or inference. Next, the proportion of labeled samples in the training set will be changed and a series of experiments will be conducted.

All signals are generated by MATLAB. The hardware CPU is Intel (R) core (TM) i7-9700, and the GPU is GeForce RTX 2070s. The running memory is 20GB.

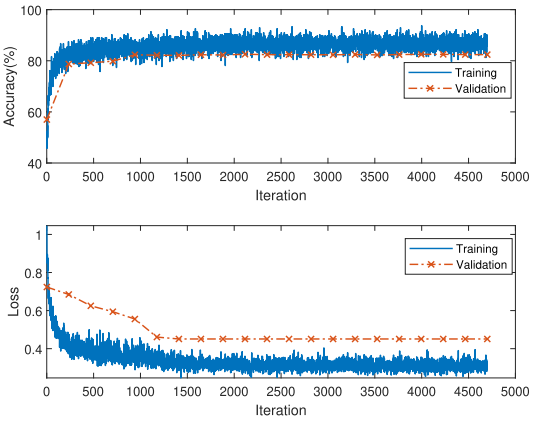
B. ALGORITHM PERFORMANCE ANALYSIS

1) THE CHOICE OF SELF-TRAINING TIMES

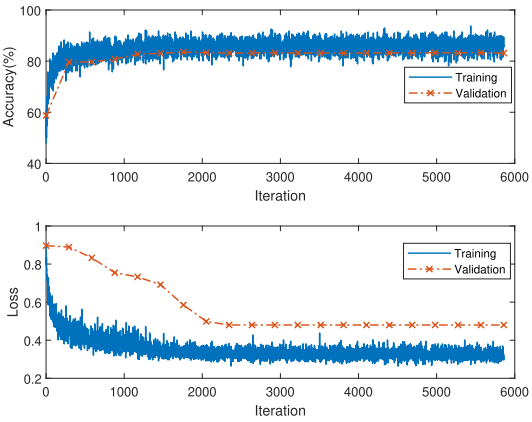
6,240 samples are selected from the training set in Section III-A as the labeled training set, accounting for 5% of the original set, and the rest of the training set are regarded as unlabeled samples. The maximum number of self-training T is set to 3, and the balance coefficient λ is 0.01. For the binary classification problem, the category with the probability of greater than 0.5 will be selected as the decision result, but the preference of model for a particular category cannot be controlled. Therefore, 15% samples are extracted from the labeled training sets to form the support set. The support set is used to test the trained classifier, and the confidence threshold is selected when the accuracy is the highest. Then the unlabeled signal samples are labeled with pseudo-labels gradually by the self-training algorithm. The curves of



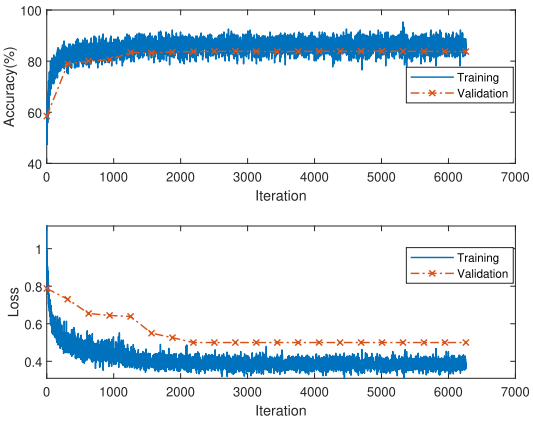
(a) Pre-training process.



(b) First self-training process.



(c) Second self-training process.



(d) Third self-training process.

FIGURE 3. **Self-training process of semi-supervised deep neural network.**

training accuracy and loss function with the iterations of the self-training algorithm are shown in Fig. 3 (a) − (d).

Figure 3 (a) shows the classification results of the classifier trained with only 6,240 labeled samples. With the increase of iterations, the classification accuracy is gradually improved, and the cross entropy loss is decreasing. However, due to the limited samples, the classification accuracy is only 73.08%. Figure 3 (b) shows the results of the training process after adding pseudo-label samples with high confidence for the first time. As more valuable information of samples is added to the classifier, the feature extraction ability of the SSDNN model for signal samples is enhanced, so the classification accuracy is improved to 82.81%. Figure 3 (c) and Figure 3 (d) show the training process after adding pseudo-label samples for the second time and the third time respectively, and the classification accuracy is 83.13%, and 83.82% respectively. Even after three times of self-training, the training accuracy is still less than 100%. This is because when the SNR is low, a large amount of signal samples will be wrongly classified as noise samples. Moreover, there are false labels in unlabeled samples with high confidence, which will affect the classification accuracy to a certain extent.

After the first self-training, it can achieve higher accuracy, and the loss function does not continue to decline significantly after increasing the number of times of self-training. As self-training will improve the training time of the model, $T = 1$ is set in the following experiments.

2) VISUALIZATION OF HIDDEN SPACE FEATURES

The feature extraction and classification of SSDNN are carried out in hidden space. Aiming to intuitively see the distribution of high-dimensional signals in the classifier, the two-dimensional data mapped by the fully connected layer in SSDNN model is visualized, as shown in Figure 4 (a) − (c), where samples belonging to H_0 are represented by "o" and samples belonging to H_1 are represented by "+". Figure 4 (a) shows the distribution of samples in two-dimensional space after just initializing the deep learning network. It can be observed that the PU signal represented by H_1 overlaps with the Gaussian white noise represented by H_0, and the

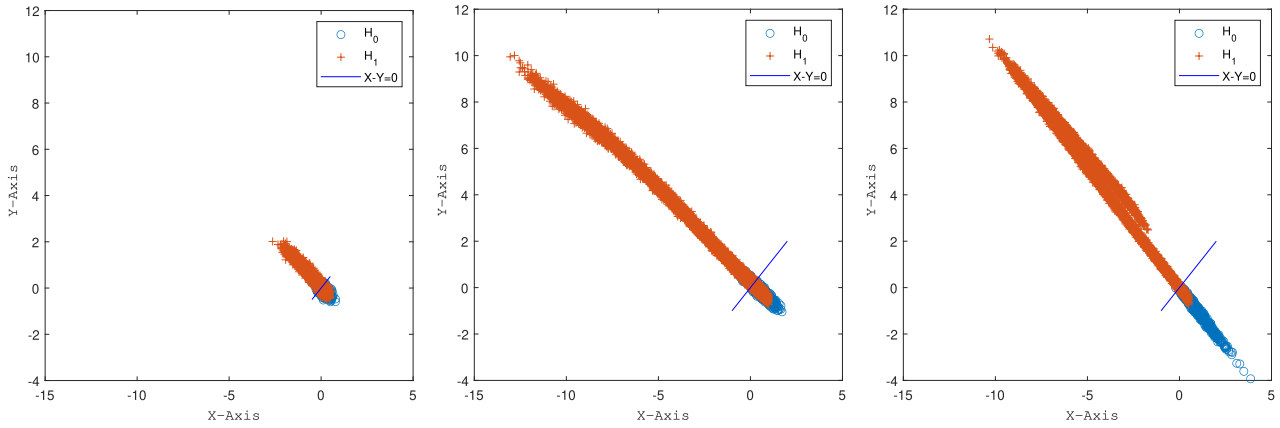


FIGURE 4. 2D visualization of deep features. (a) accuracy = 51.2%, after just initializing the deep learning network. (b) accuracy = 74.6%, after the pre-training phase. (c) accuracy = 83.3%, after the semi-supervised learning phase and the training and sensing phase.

category of each sample cannot be distinguished from the visual diagram alone. Figure 4 (b) shows the spatial distribution of two-dimensional signals with the classification accuracy of 74.6% after the pre-training phase. Compared with Fig. 4 (a), the two categories of samples is of zonal distribution, and the differentiation is apparent, but there are still some overlapping areas. Figure 4 (c) shows the spatial distribution of two-dimensional signals with a classification accuracy of 83.3% after the semi-supervised learning phase and the training and sensing phase. Except for a small part of the overlap, H_0 or H_1 can be distinguished as a whole. At the same time, it can be seen from Fig. 4 (a) − (c) that the noise distribution is more concentrated. In contrast, the signal distribution is loose, which may be contributed to the information of different modulation types in mapped two-dimensional signal features.

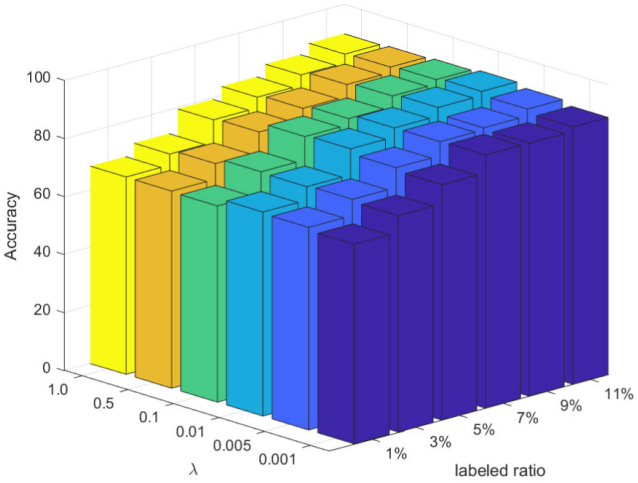


FIGURE 5. Accuracy with different λ.

3) INFLUENCE OF BALANCE COEFFICIENT ON ALGORITHM PERFORMANCE

To explore the influence of balance coefficient λ on classification accuracy, the proportion of labeled samples to the original set is 1%, 3%, 5%, 7%, 9%, and 11% respectively. The relationship between the accuracy of the SSDNN and the value of λ in different proportions of labeled samples is shown in Figure 5. From the figure, some conclusions can be obtained. (1) The balance coefficient has an impact on the accuracy of SSDNN algorithm. (2) When the balance coefficient λ is fixed, increasing the proportion of labeled samples can improve the classification accuracy on the whole. This is because when the number of labeled samples is small, the fitting ability of the SSDNN model is not strong. With the increase of the proportion of labeled samples in the total samples, the fitting ability is enhanced, and the different features of the signal can be extracted better. Especially when the proportion of labeled samples increases from 1% to 7%, the classification accuracy can be significantly improved. (3) When the number of labeled samples is fixed, the classification accuracy does not always increase with the increase of λ, since the increase of λ means that the impact of pseudo labeled samples on the training model increases. If the proportion of false labeled samples is large, the model will be trained in the wrong direction. In addition, according to the experimental results shown in Table 2, higher classification accuracy can be obtained when the value of λ is between 0.005 and 0.1. Taking $\lambda = 0.01$ as an example, when the proportion of labeled samples increases from 1% to 5%, the accuracy is improved by 6.6% and 11.9% respectively, which shows the effectiveness of the SSDNN when the proportion of labeled samples is low.

4) INFLUENCE OF CONFIDENCE THRESHOLD P ON ALGORITHM PERFORMANCE

In order to study the influence of confidence threshold p on classification accuracy, the values of p are set to 0,0.2,04,0.6,0.8 and 1 respectively in 5% labeled sample

TABLE 2. Labeling accuracy with different λ.

λ	Accuracy(%)					
	1%	3%	5%	7%	9%	11%
0.001	68.72	74.49	81.08	**87.20**	87.12	88.62
0.005	69.81	**75.36**	82.16	86.96	87.62	89.98
0.01	**70.28**	74.92	**83.82**	86.92	**89.84**	**91.26**
0.1	67.62	75.24	83.26	85.32	89.02	90.54
0.5	67.93	73.28	80.02	84.08	88.16	90.12
1.0	67.99	71.78	79.46	83.12	86.70	89.78

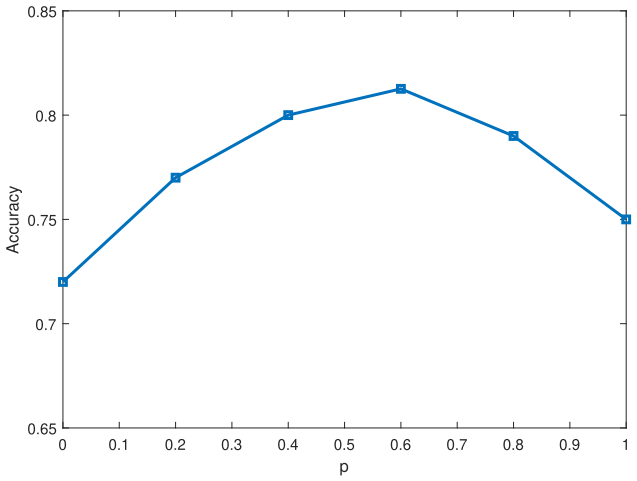


FIGURE 6. Accuracy with different p.

dataset. The balance coefficient is set to 0.01. For simplicity, the number of self-training $T = 1$ is assumed. The relationship between the accuracy of SSDNN algorithm and p value under different confidence thresholds is shown in Figure 6. As can be seen from the figure: (1) the maximum accuracy can be obtained when the value of p is 0.6. According to equation 17, p equals to 0.6 means that a certain category is selected and marked with a pseudo-label when its probability is greater than 0.8. (2) p equals to 1 means that a certain category is selected and marked with a pseudo-label when its probability is 1. In this case, only few samples will be selected and marked. The performance of the classifier mainly depends on 5% of the labeled samples. Because the number of labeled samples is too small, the classification accuracy is not ideal. (3) When the p is 0, it means that the probability of a certain category is greater than 0.5 before it is selected and marked. In this case, all unlabeled samples will be marked. As there are wrong labeled samples in the dataset, the classification accuracy is also not high.

5) INFLUENCE OF SIGNAL SAMPLING LENGTH *N* ON ALGORITHM PERFORMANCE

In order to study the effect of the length of the signal sample N on the probability of detection, the values of N are set to 512, 1024, and 2048 respectively in 5% labeled-sample dataset to conduct experiments. The balance coefficient is set to 0.01 and the false alarm probability is set to 0.1 by

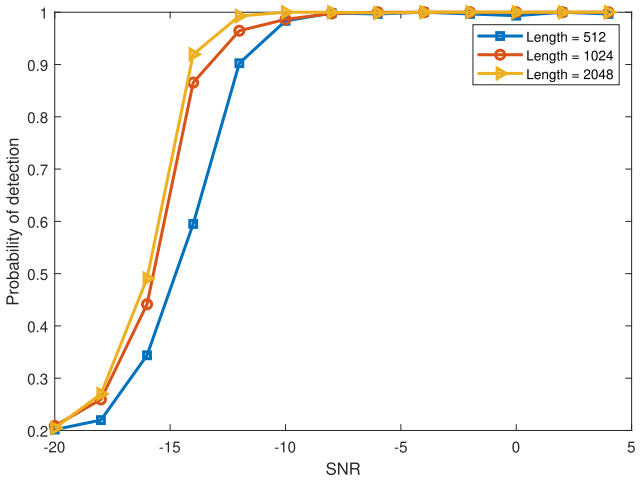


FIGURE 7. P_d under different signal lengths.

adjusting γ, which is required in the IEEE 802.22 standard. It can be seen from the Figure 7 that the detection probability increases with the increase of the length of the signal sample. Especially when the length increases from 512 to 1024, the signal sample contains more useful information, and it is used by the classifier, so the detection probability of the algorithm increases significantly. However, when the signal sample length increases from 1024 to 2048, the improvement of detection probability is not obvious at the range between −20dB and −14dB. This is because when the SNR is low and the classification performance of the trained classifier is limited, and blindly increasing the signal sample length cannot effectively improve the perceptual performance. (2) The parameters and complexity of the model increase with the increase of signal sample length. When the detection probability is 95% and the length are 2048, 1024 and 512, respectively, the SNR needed are −13.6dB, −12.5dB, and −11dB, respectively. The longer the sample length is, the higher the complexity of the algorithm is. Therefore, considering the performance and complexity of the algorithm, the sample length $N = 1024$ is taken latter.

6) INFLUENCE OF DIFFERENT FALSE ALARM PROBABILITY P_f ON ALGORITHM PERFORMANCE

In order to study the effect of the false alarm probability P_f on the detection performance, the value of P_f is set to 0.1, 0.05, and 0.01 respectively to conduct experiments. The proportion of labeled samples is set to 5% and 10% respectively. It can be observed from the figure 8 that when the false alarm probability is fixed, the detection probability increases with the number of labeled samples. For example, when $P_f = 0.1$ and SNR is −15 dB, the detection probability of 5% and 10% labeled samples are 71% and 98% respectively. At the same time, when the detection probability is 100% and the proportion of labeled samples is 10%, under the three false alarm probabilities, the required SNR are −12 dB, −10 dB, and −8 dB, respectively.

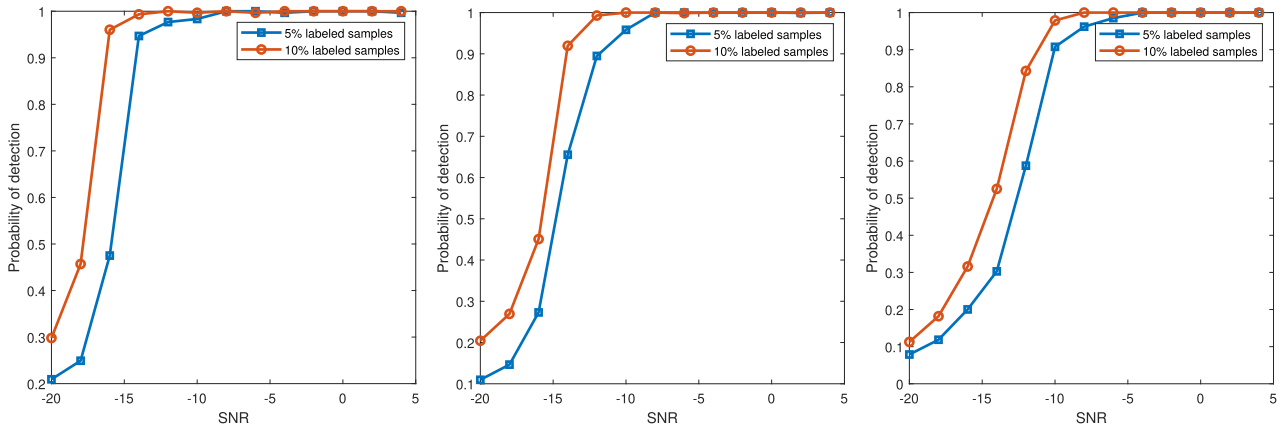


FIGURE 8. P_d under different P_f. (a) $P_f = 0.1$, (b) $P_f = 0.05$ and (c) $P_f = 0.01$.

7) INFLUENCE OF CHANNEL FADING ON ALGORITHM PERFORMANCE

To explore the detection performance of SSDNN in different channel fading scenarios, four scenarios are considered: no fading, Rayleigh fading, Rican fading and Nakagami Fading. Figure 9 shows the simulation results of SSDNN in 5% labeled samples, where P_f is set to 0.05, $\lambda = 0.01$. The three path delays and path gains of Rayleigh fading, Rican fading and Nakagami fading are [0, 4.5, 8.5]*e-05 and [0, −2, −10]dB respectively. As can be seen from the figure, different fading channels impact the spectrum sensing performance of the SSDNN. In the Gaussian channel, the SSDNN algorithm has the highest detection performance, and in the Rayleigh channel, the SSDNN algorithm has the worst detection performance. For example, when the detection probability is 90%, the needed SNR of the proposed algorithm in Rayleigh fading is −9dB, which is 1.5 dB, 2.8 dB and 3.5 dB higher than that in Nakagami Fading, Rican Fading and no fading scenarios, respectively.

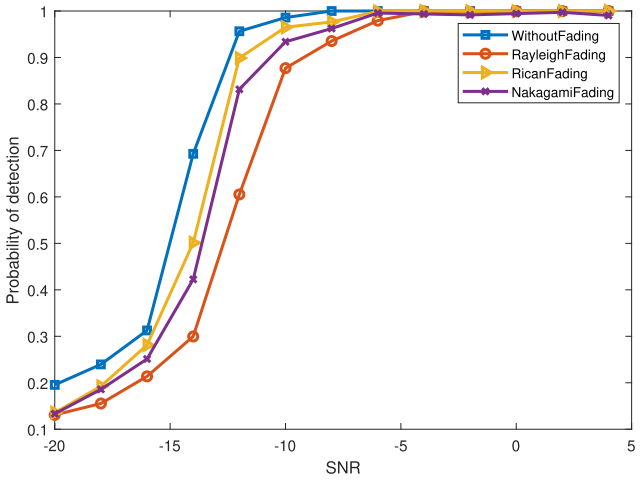


FIGURE 9. P_d under different fadings.

8) INFLUENCE OF FREQUENCY OFFSET ON ALGORITHM PERFORMANCE

In order to illustrate the influence of carrier frequency offset on the SSDNN method, the sensing performance on different frequency offsets is analyzed. Energy detection method (ED) [6], entropy-based spectrum sensing method (Entropy) [9] and supervised learning CNN [21] are compared. When there is a frequency offset, the received signal can be described as:

$$x(n) = e^{j2\pi \Delta f_c n/f_c} h(n)s(n) + v(n) \tag{21}$$

wherein Δf_c is the center frequency offset of the transmitted signal and the received signal, and $\Delta f_c/f_c$ is the normalized carrier frequency offset. In the experiment, P_f is set to 0.05; the sampling length N equals 2048; 5% labeled samples constitute the training sets, and $\Delta f_c/f_c$ is set to −0.1, 0 and 0.1. The sensing results of four algorithms are shown in Figure 10. As can be seen from the figure, the performance curves of the SSDNN and CNN [21] hardly change with the deviation of carrier frequency, so the two methods can resist the frequency offset. The frequency offset has a little effect on the traditional ED and Entropy methods, and their performance of resisting the frequency offset is worse than the former two. Both SSDNN and CNN are based on the deep learning model, in which only the original IQ data is input. The results show that the deep learning methods can learn the features of frequency offset in the training process.

9) PERFORMANCE COMPARISON

This section compares the performance of SSDNN, energy detection method (ED) [6], entropy-based spectrum sensing method (Entropy) [9] and supervised learning CNN [21]. The labeled sample proportions of SSDNN are 5%, 10% and 100% respectively, and P_f is 0.1, $\lambda = 0.01$. The detection probability curves of the four algorithms are shown in Figure 11. It can be observed from the figure that the

TABLE 3. Computational complexity of different methods.

Method	Offline Training	Online Detection
ED	–	$O(N)$
Entropy	–	$O(N)$
CNN	$O\left(N_t N_e \sum_{l=1}^{D} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$	$O\left(\sum_{l=1}^{D} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$
SSDNN	$O\left((T+1) N_t N_e \sum_{l=1}^{D} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$	$O\left(\sum_{l=1}^{D} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$

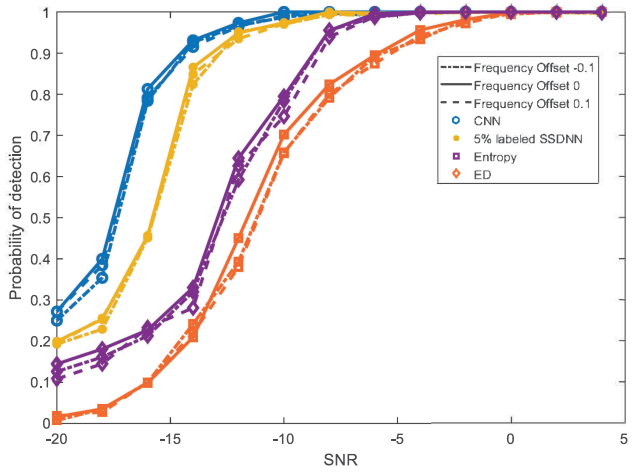


FIGURE 10. P_d under different frequency offsets.

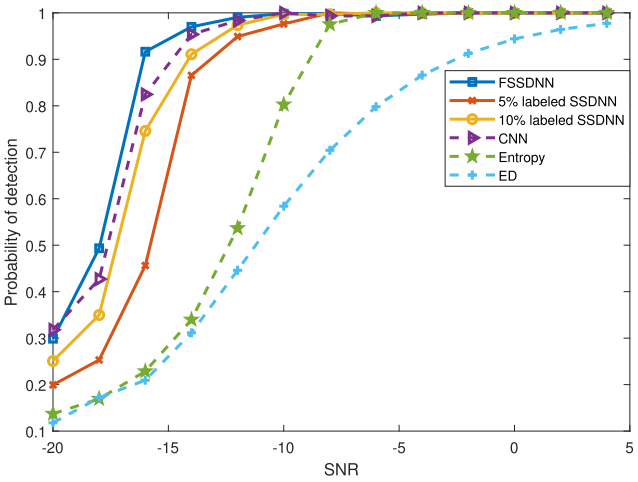


FIGURE 11. Comparison of different methods.

spectrum sensing methods based on deep learning are better than the traditional spectrum sensing methods (ED and Entropy), owing to the ability that deep learning can learn the characteristics of PU signal and noise from a quantity of samples. When the detection probability is 90%, the needed SNR of fully labeled SSDNN(abbreviated as FSSDNN), CNN [21], 10% labeled SSDNN and 5% labeled SSDNN are −16.2 dB, −14.9 dB, −14.1 dB and −13.2 dB respectively. The performance of deep learning spectrum sensing algorithms with fully labeled samples (FSSDNN and CNN) is better than that of SSDNN. This is because the former is a fully supervised learning method with sufficient information of labeled samples. Still, only the samples with high confidence will be marked in SSDNN, which will reduce the total number of samples in training. However, the sensing performance of 10% labeled samples of the SSDNN method is close to that of the CNN method.

10) COMPLEXITY ANALYSIS

For fair comparison, the computational complexity of the four methods is analyzed. The computational complexity of the four algorithms is summarized in Table 3. Both SSDNN and CNN algorithms need to establish models through offline training, while entropy and ED algorithms do not have training process and can directly carry out online detection. Among them, entropy and ED algorithms do not involve matrix operation, and their complexity is $O(N)$. It can be seen from reference [40] that the time complexity of calculating all convolution layers is $O\left(\sum_{l=1}^{D} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$, Where l is the index of a convolutional layer, D is the number of convolutional layers, n_l is the number of filters in the lth layer, s_l is the spatial length of the filter and m_l is the spatial size of the output feature map. In addition, N_t and N_e denote the numbers of training samples and epochs of the offline training, respectively. For CNN, an extra offline training is required, and its main complexity depends on N_t and N_e. The offline training of SSDNN mainly includes two stages, that is, pre-training and self-training. The computational complexity of pre-training is the same as that of CNN. Setting the number of self-training as T, the total computational complexity is shown in Table 3. It should be noted that in the pre-training stage of SSDNN, only small amounts of the N_t samples with true-label participate in the training. Similarly, in the self-training stage, only some of the N_t samples with true-label and pseudo-label participate in the training. Therefore, in fact, the computation time of SSDNN algorithm is less than $T+1$ times that of CNN. This is also illustrated by the actual runtime of the four algorithms in Table 4. It can also be seen from the table that the traditional spectrum sensing methods can realize spectrum sensing in a short time, while the spectrum sensing methods based on deep learning need a lot of training time due to the complex model, but the accuracy of the methods based on deep learning is higher than that of the traditional method. Although the SSDNN has the highest training time complexity, its online

TABLE 4. Runtime of different methods.

Method	Offline Training(s)	Online Detection(ms)
ED	–	0.075
Entropy	–	0.092
CNN	2435	0.172
SSDNN	3487	0.158

detection time is lower than that of CNN once the model is well-trained.

IV. CONCLUSION

In order to tackle the issue that it is challenging to obtain vast amounts of labeled samples in real radio environment, this paper provides a limited data spectrum sensing method based on semi-supervised deep neural network (SSDNN). The SSDNN does not require any prior information about signal and noise distributions. Compared with the existing supervised learning based spectrum sensing algorithms, the SSDNN requires a smaller number of labeled training set, which effectively solves the problem of insufficient labeled samples in practical application, and provides a new solution for spectrum sensing with limited samples. The performances of the SSDNN algorithm in different lengths, different false alarm probabilities, different fading scenarios and different frequency offset scenarios are simulated and analyzed for a dataset of 124,800 samples. It can be observed via the results of the simulations that when the labeled samples account for only 10% of the traditional deep learning networks such as CNN, the classification accuracy of the SSDNN is close to that of CNN.

REFERENCES

[1] P. Kolodzy, "Spectrum policy task force report," *IEEE Trans. Inf. Forensics Security*, vol. 40, no. 4, pp. 147–158, Nov. 2002.
[2] J. Mitola and G. Q. Maguire, Jr., "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, no. 4, pp. 13–18, Apr. 1999.
[3] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Commun.*, vol. 14, no. 4, pp. 47–52, Aug. 2007.
[4] T. Xiong, Y.-D. Yao, Y. Ren, and Z. Li, "Multiband spectrum sensing in cognitive radio networks with secondary user hardware limitation: Random and adaptive spectrum sensing strategies," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3018–3029, May 2018.
[5] O. M. Eltabie, M. F. Abdelkader, and A. M. Ghuniem, "Incorporating primary occupancy patterns in compressive spectrum sensing," *IEEE Access*, vol. 7, pp. 29096–29106, 2019.
[6] B. Gaiera, D. K. Patel, B. Soni, and M. Lopez-Benitez, "Performance evaluation of improved energy detection under signal and noise uncertainties in cognitive radio networks," in *Proc. IEEE Int. Conf. Signals Syst. (ICSigSys)*, Jul. 2019, pp. 131–137.
[7] J. C. Shen and E. Alsusa, "An efficient multiple lags selection method for cyclostationary feature based spectrum-sensing," *IEEE Signal Process. Lett.*, vol. 20, no. 2, pp. 133–136, Feb. 2013.
[8] X. Zhang, F. Gao, R. Chai, and T. Jiang, "Matched filter based spectrum sensing when primary user has multiple power levels," *China Commun.*, vol. 12, no. 2, pp. 21–31, Feb. 2015.
[9] Y. Zhang, Q. Zhang, and S. Wu, "Entropy-based robust spectrum sensing in cognitive radio," *IET Commun.*, vol. 4, no. 4, pp. 428–436, 2010.
[10] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio networks," in *Proc. 1st IEEE Int. Symp. New Frontiers Dyn. Spectr. Access Netw. (DySPAN*, Jan. 2005, pp. 137–143.
[11] Y. Lv, Z. Song, and Y. Liu, "Simultaneous cooperative spectrum sensing and wireless power transfer in single-antenna cognitive radio," in *Proc. IEEE 4th Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Dec. 2019, pp. 899–903.
[12] R. Liu, Y. Ma, X. Zhang, and Y. Gao, "Deep learning-based spectrum sensing in space-air-ground integrated networks," *J. Commun. Inf. Netw.*, vol. 6, no. 1, pp. 82–90, Mar. 2021.
[13] J. Xie, J. Fang, C. Liu, and L. Yang, "Unsupervised deep spectrum sensing: A variational auto-encoder based approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5307–5319, May 2020.
[14] H. He and H. Jiang, "Deep learning based energy efficiency optimization for distributed cooperative spectrum sensing," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 32–39, Jun. 2019.
[15] B. M. Pati, M. Kaneko, and A. Taparugssanagorn, "A deep convolutional neural network based transfer learning method for non-cooperative spectrum sensing," *IEEE Access*, vol. 8, pp. 164529–164545, 2020.
[16] J. Gao, X. Yi, C. Zhong, X. Chen, and Z. Zhang, "Deep learning for spectrum sensing," *IEEE Wireless Commun. Lett.*, vol. 8, no. 6, pp. 1727–1730, Dec. 2019.
[17] T. J. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conf.*, vol. 1, Sep. 2016, pp. 1–6.
[18] G. Pan, J. Li, and F. Lin, "A cognitive radio spectrum sensing method for an OFDM signal based on deep learning and cycle spectrum," *Int. J. Digit. Multimedia Broadcast.*, vol. 2020, pp. 1–10, Mar. 2020.
[19] J. Xie, C. Liu, Y.-C. Liang, and J. Fang, "Activity pattern aware spectrum sensing: A CNN-based deep learning approach," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 1025–1028, Jun. 2019.
[20] M. Ozdes and B. M. Severoglu, "Sound spectrum detection using deep learning," in *Proc. Sci. Meeting Elect.-Electron. Biomed. Eng. Comput. Sci. (EBBT)*, Apr. 2019, pp. 1–4.
[21] S. Zheng, S. Chen, P. Qi, H. Zhou, and X. Yang, "Spectrum sensing based on deep learning classification for cognitive radios," *China Commun.*, vol. 17, no. 2, pp. 138–148, Feb. 2020.
[22] D. Chew and A. B. Cooper, "Spectrum sensing in interference and noise using deep learning," in *Proc. 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020, pp. 1–6.
[23] C. Liu, J. Wang, X. Liu, and Y.-C. Liang, "Deep CM-CNN for spectrum sensing in cognitive radio," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2306–2321, Oct. 2019.
[24] J. Xie, J. Fang, C. Liu, and X. Li, "Deep learning-based spectrum sensing in cognitive radio: A CNN-LSTM approach," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2196–2200, Oct. 2020.
[25] A. Subekti, H. F. Pardede, and R. Sustika, "Spectrum sensing for cognitive radio using deep autoencoder neural network and SVM," in *Proc. Int. Conf. Radar, Antenna, Microw., Electron., Telecommun. (ICRAMET)*, Nov. 2018, pp. 81–85.
[26] H. Zhou, L. Jiao, S. Zheng, L. Yang, W. Shen, and X. Yang, "Generative adversarial network-based electromagnetic signal classification: A semi-supervised learning framework," *China Commun.*, vol. 17, no. 10, pp. 157–169, Oct. 2020.
[27] Y. Hu, R. Liu, X. Li, D. Chen, and Q. Hu, "Task-sequencing meta learning for intelligent few-shot fault diagnosis with limited data," *IEEE Trans. Ind. Informat.*, early access, Sep. 14, 2021, doi: 10.1109/TII.2021.3112504.
[28] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 2008.
[29] H. Yu, C. Sun, X. Yang, S. Zheng, and H. Zou, "Fuzzy support vector machine with relative density information for classifying imbalanced data," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 12, pp. 2353–2367, Dec. 2019.
[30] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 4, pp. 580–585, Aug. 1985.
[31] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit.*, vol. 2, Aug. 2004, pp. 28–31.
[32] A. De Angelis, G. De Angelis, and P. Carbone, "Using Gaussian-uniform mixture models for robust time-interval measurement," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 12, pp. 3545–3554, Dec. 2015.

[33] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar. 1996.
[34] S. Martin and C. T. M. Choi, "Nonlinear electrical impedance tomography reconstruction using artificial neural networks and particle swarm optimization," *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, Mar. 2016.
[35] O. I. Abiodun, M. U. Kiru, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, and U. Gana, "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE Access*, vol. 7, pp. 158820–158846, 2019.
[36] M. Narendar, A. P. Vinod, A. S. Madhukumar, and A. K. Krishna, "A robust two-stage spectrum sensing method using filter bank based energy detector and fourth order cumulants for cognitive radios," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, Nov. 2012, pp. 438–442.
[37] T. Liu, Y. Yang, G. B. Huang, Y. K. Yeo, and Z. Lin, "Driver distraction detection using semi-supervised machine learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1108–1120, Apr. 2016.
[38] Z. Zhang, M. Zhao, and T. W. S. Chow, "Graph based constrained semi-supervised learning framework via label propagation over adaptive neighborhood," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2362–2376, Sep. 2015.
[39] X. Lu, J. Zhang, T. Li, and Y. Zhang, "A novel synergetic classification approach for hyperspectral and panchromatic images based on self-learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4917–4928, Aug. 2016.
[40] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5353–5360.

YUPEI ZHANG received the B.S. and M.S. degrees from the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, China, in 2016 and 2019, respectively, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information. His current research interests include deep learning and cognitive radio.

ZHIJIN ZHAO received the M.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 1984 and 2009, respectively. In 1993 and 2003, as a Visiting Scholar, she studied adaptive signal processing and blind signal processing with the Darmstadta University of Technology and the University of Erlangen-Nuremberg, respectively. She is currently a Professor with the School of Communication Engineering, Hangzhou Dianzi University. She once worked as the President of the College of Communication Engineering, and a Senior Member of China Electronics Society, a member of National Signal Processing Society, the Secretary General of Zhejiang Signal Processing Society, and the Director of Hangzhou Electronics Society. Her research interests include communication signal processing, cognitive radio technology, intelligent signal processing, and other aspects of research.