Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 19, No. 1, pp. 161–173, 2023

SEMI-SUPERVISED ADVERSARIAL AUTOENCODER BASED SPECTRUM SENSING FOR COGNITIVE RADIO*

Wei Liang, Xi Zhang, Jianhua Yuan, Caixia Kou and Wenbao Ai†

Abstract: Spectrum sensing is a basic supporting technology of cognitive radio. Most of the existing detection methods are based on the energy method and other methods, which can be simply implemented, but the detection performances are poor in the case of low signal-to-noise ratio (SNR). In order to resolve this issue, many researchers have introduced deep learning into spectrum sensing and obtained good results, but most of the algorithms are supervised learning, which require a lot of labeled training samples. In this paper, we propose a semi-supervised spectrum sensing algorithm, which is based on the semi-supervised adversarial autoencoder (SSAAE) model. Compared with the supervised algorithms, our algorithm requires only a small amount of labeled samples. Moreover, it does not require any prior information and can directly use the received signals as the network input without any preprocessing. The simulation experiment results show that the detection probability of our proposed algorithm can reach 0.9216 when SNR and the false alarm probability are −14dB and 0.1 respectively, which is much higher than classic methods and some deep learning-based methods.

Key words: *cognitive radio, spectrum sensing, deep learning, adversarial autoencoder*

Mathematics Subject Classification: *68T07, 94-10, 94A16*

1 Introduction

In order to solve the contradiction between spectrum shortage and low utilization, cognitive radio (CR) is proposed to effectively use spectrum in different time and space [13]. Cognitive radio allows secondary users (SUs) to access the spectrum of authorized primary users (PUs) [10]. Cognitive radio receivers periodically receive target frequency band information and decide whether the target frequency band is idle through analysis. If it is in the idle phase, the corresponding SU can quickly access the frequency band. Otherwise, it starts to scan other frequency bands. Spectrum sensing is the basic technology of cognitive radio [1], which means to obtain the spectrum usage of target frequency band through various signal processing methods.

So far, many spectrum sensing methods have been proposed, such as cyclostationary feature detection (CFD) [20], energy detection (ED) [4], the maximum eigenvalue detection (MED) [22], and covariance absolute value (CAV) detection [23]. CFD mainly determines the presence or absence of the PU based on the cyclic stationary statistics or the cyclic

*This work was supported by the National Natural Science Foundations of China (No.11871115, 12171052, 11971073, 12171051). This work was also supported by Beijing Natural Science Foundation (No. Z220004).
†Corresponding author.

frequency of the received signals, so it requires prior information of the signals. ED uses the energy of the signals in a specific time to make judgment, which is simply implemented and widely used, but does not take effect at low SNR. MED uses the correlation of the signals to calculate the eigenvalue of the covariance matrix of the signals. When the signals are highly correlated with each other, MED can get good results even in the case of low SNR. However, ED and MED need to know noise power, so that they are semi-blind detection methods; moreover, their actual detection performances are worse than theoretical performances due to noise uncertainty [15]. The CAV detection does not require any prior information, which belongs to the blind detection method. What's more, this method will not be affected by noise uncertainty. But it is difficult to determine the detection threshold accurately under the non-progressive assumption.

With the advent of the era of artificial intelligence and big data, deep learning has achieved good applications in image processing [3], natural language processing [21], sentiment analysis [5] and other fields. Of course, these methods have also been applied to spectrum sensing. In [17], Vyas et al. proposed a hybrid spectrum sensing scheme based on artificial neural network (ANN), which used energy values and likelihood ratio statistics as the input of the network; In [16], Tang et al. combined time-domain and frequency-domain information and put forward to use energy and cyclostationary features as the input of ANN; In [9]−[19], the authors used the covariance matrix as the input and trained it with the classical convolutional neural network (CNN) model. And Liu et al. [9] derived the deep-neural-network-based likelihood ratio test (DNN-LRT) for spectrum sensing; In [2], Cheng et al. used a stacked autoencoder to process the signal samples and logistic regression to classify user activities. Overall, all of the above algorithms belong to supervised deep learning, that is, they need to know the labels of training samples. But in actual situations, collecting large amounts of labeled training samples is costly and difficult. In order to break through this limitation, Xie et al. [18] proposed an unsupervised deep learning spectrum sensing algorithm (UDSS), which used a variational autoencoder (VAE) [7] to extract features from the sample covariance matrix, then clustered features by means of Gaussian mixture model (GMM) [24], and finally used a small amount of label samples that the PU does not presence for cluster identification.

In this paper, we present a spectrum sensing algorithm based on SSAAE. The adversarial autoencoder (AAE) [11] combines a generative adversarial network (GAN) [6] and a variational autoencoder (VAE). Although both AAE and VAE match the posterior distribution with the prior distribution through certain constraints, their process of matching is different. The former uses adversarial training and the latter uses Kullback-Leibler divergence. AAE has a wide range of applications, such as semi-supervised classification, style separation, unsupervised clustering, data dimension reduction, etc. In [12], Sudhanshu Mittal researched on semi-supervised learning of AAE and did simulation experiments on mixed national institute of standards and technology (MNIST) dataset, and obtained some good results. In SSAAE model, the label information is added to the AAE model for semi-supervised classification tasks. The SSAAE model tries to utilize unlabeled samples to improve the classification performance, which not only solves the shortage of labeled samples, but also improves the generalization ability of the model.

The main contributions of this paper can be summarized as follows:

- We propose an SSAAE-based spectrum sensing algorithm, which is a semi-supervised classification algorithm. So in the training phase, we use only a small number of labeled samples. As we know, it is the first time that the SSAAE model is applied to spectrum sensing.

- In this algorithm, we use only the original received signals as the network input, that is, we do not require other feature extraction methods. In addition, the algorithm does not require any prior information about the signals and the noises. So it belongs to blind detection methods.
- The proposed algorithm is simulated under white Gaussian noise and compared with classical spectrum sensing methods. Experiments show that the proposed algorithm is superior to classic methods (ED, MED and CAV) and two deep learning-based methods (ANN-based and CM-CNN-based method).

The remainder of this paper is organized as follows. In Section 2, the binary classification model of spectrum sensing is described. The idea and structure of the SSAAE model are introduced in Section 3. The spectrum sensing algorithm based on SSAAE is given in Section 4. In Section 5, the performance of the proposed algorithm is showed by simulation experiments. A conclusion is presented in Section 6.

2 Formulation of the Problem

We assume that the SU has M receiving antennas, and the received signals are sampled N times for each sensing period. Let $x_m(n) \in \mathbb{C}$ $(m = 1, \ldots, M, n = 1, \ldots, N)$ denote the n-th sampled signal received from the m-th antenna by the SU. If the primary user does not presence in the target channel, each the received signal $\{x_m(n)\}$ contains only one white noise $v_m(n) \in \mathbb{C}$, that is $x_m(n) = v_m(n)$; otherwise it is equal to the sum of a signal $s_m(n) \in \mathbb{C}$ sent by the PU after channel fading and the white noise $v_m(n)$, that is $x_m(n) = s_m(n) + v_m(n)$. Spectrum sensing is that the SU needs to detect, by the received signals $\{x_m(n)\}$, whether the PU is present in the target channel or not. Let H_0 and H_1 represent the absence and presence of the PU in the target channel, respectively. The spectrum sensing problem is just a binary hypothesis testing problem:

$$\begin{aligned} H_0 &: \ x_m(n) = v_m(n), \\ H_1 &: \ x_m(n) = s_m(n) + v_m(n). \end{aligned} \tag{2.1}$$

The performance of spectrum sensing can be portrayed by a detection probability P_d and a false alarm probability P_f, which are respectively defined as:

$$P_d = P\{H_*|H_1\}, \ P_f = P\{H_*|H_0\}, \tag{2.2}$$

where H_* represents that the SU has detected the PU would be present in the target channel. Obviously, we want the detection probability P_d to be as high as possible, and the false alarm probability P_f to be as low as possible. Therefore, the key task of spectrum sensing is to maximize detection probability P_d under a given tolerance of the false alarm probability P_f:

$$\begin{aligned} max \quad & P_d \\ & P_f \leq \Delta, \end{aligned} \tag{2.3}$$

where Δ represents a given threshold of P_f.

3 The Semi-Supervised Adversarial Autoencoder

As shown in Fig. 3.1, the SSAAE model consists of three parts: an autoencoder in the middle part, a categorical discriminator in the top part and a Gaussian distribution discriminator in bottom part.

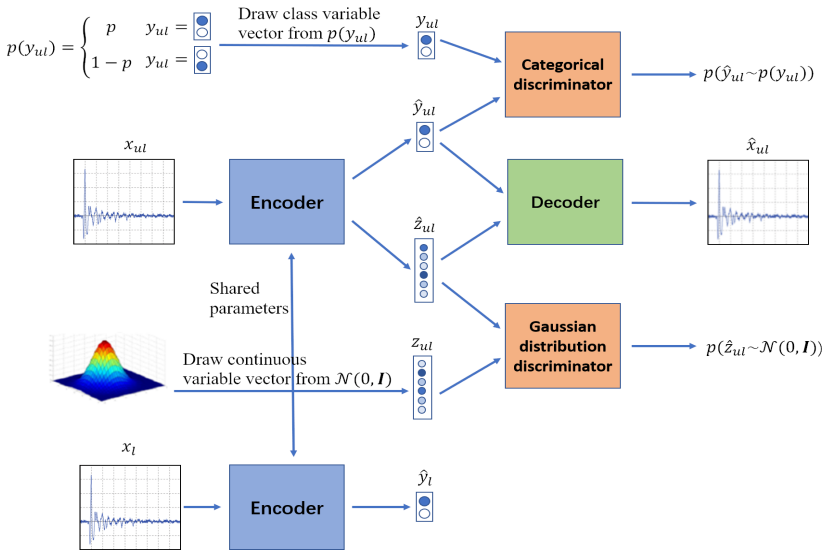

Figure 3.1: Architecture of a semi-supervised adversarial autoencoder [11]. $p(\hat{y}_{ul} \sim p(y_{ul}))$ denotes the probability that the categorical discriminator determines that $\hat{y}_{ul}$ follows $p(y_{ul})$ distribution, and $p(\hat{z}_{ul} \sim \mathcal{N}(0, \mathbf{I}))$ denotes the probability that the Gaussian distribution discriminator determines that $\hat{z}_{ul}$ follows the Gaussian distribution.

Let x denote an input vector of the network (x includes x_l and x_{ul}, which come from labeled samples and unlabeled samples, respectively), and z_{ul} be a randomly generated continuous variable vector subject to the Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. In addition, let y be a two-dimensional class variable vector. It takes two values: $y = [1, 0]^T$ or $[0, 1]^T$. The former value claims that the PU is absent, that is the current x consists only of noises, while the latter asserts that x includes signals of the PU. And the entire training process of SSAAE model includes one supervised phase and one unsupervised phase. In the supervised phase, y identifies really whether the current x_l contains signals of the PU or not, which will be denoted as y_l in remaining sections. However, for the unsupervised phases, since the label of the input x_{ul} is unclear, y takes stochastically the two values $[1, 0]^T$ and $[0, 1]^T$ with probabilities p and $1 - p$ respectively, where p is the proportion of the pure noise samples in all the unlabeled samples. In this case, the vector y will be denoted as y_{ul}. So, it may not reflect the true label of the input sample x_{ul}.

The autoencoder consists of an encoder and a decoder. The encoder is a three-layer neural network. When a sample x_{ul} is input into the encoder, its outputs will be two vectors: one is a latent class variable vector $\hat{y}_{ul}$, and the other is a latent style variable vector $\hat{z}_{ul}$. In addition, the encoder contains multiple activation functions: the activation functions of the first two layers are both the rectified linear unit (ReLU) functions. In the third layer, $\hat{z}_{ul}$ is output directly through a linear transformation, while $\hat{y}_{ul}$ is output by a linear transformation and a Softmax activation function. The latent class variable $\hat{y}_{ul}$ can obtain label information, which is used in classification tasks. The latent style variable $\hat{z}_{ul}$ can obtain style information, forcing the outputs of the encoder to carry more information. Since the network parameters of the first two layers corresponding to the two outputs are shared, the training of $\hat{z}_{ul}$ will also affect $\hat{y}_{ul}$ during back propagation, which will improve the classification performance of the SSAAE model. Then, $\hat{y}_{ul}$ and $\hat{z}_{ul}$ are combined as one

input to the decoder, and the output of the decoder is a reconstructed sample $\hat{x}_{ul}$. This decoder is also a three-layer neural network. The activation functions of its first two layers are still both ReLU functions, and the last output layer uses a Tanh activation function.

The encoder and categorical discriminator together constitute the upper adversarial network, in which the encoder plays the role of generator. Any class variable vector y_{ul} is regarded as a real sample labeled by "true"; and any latent class variable vector $\hat{y}_{ul}$ is regarded as a generated sample labeled by "false". Then y_{ul} and $\hat{y}_{ul}$ are input into the categorical discriminator, which is a three-layer neural network. Its first two layers use the leaky ReLU activation function, and its last layer uses the Sigmoid activation function. The categorical discriminator learns differences between real samples and generated samples to identify them. At the same time, the encoder learns from training also, to try to make the categorical discriminator regard the generated samples as labeled "true" as possible. The output of the categorical discriminator is a probability of the current sample (y_{ul} or $\hat{y}_{ul}$) being a real sample inferred by the categorical discriminator. The upper adversarial network try to have the aggregated posterior distribution of $\hat{y}_{ul}$ to match the categorical distribution as far as possible.

Similarly, the encoder and Gaussian distribution discriminator constitute the lower adversarial network to train the encoder. Any continuous variable vector z_{ul} is regarded as a real sample labeled by "true"; and any latent style variable vector $\hat{z}_{ul}$ is regarded as a generated sample labeled by "false". The network structure of Gaussian distribution discriminator is the same as that of categorical discriminator. The lower adversarial network imposes a Gaussian distribution on the style representation which ensures the latent variable $\hat{z}_{ul}$ obeys Gaussian distribution as far as possible.

The training of the SSAAE model consists of three phases: a reconstruction phase, a regularization phase and a semi-supervised classification phase. In the reconstruction phase, the autoencoder is used to reconstruct the input sample x_{ul}, and we hope that the output of the decoder can perfectly or approximately restore the original input. In the regularization phase, two adversarial networks are used to impose prior distributions on two latent variables $\hat{y}_{ul}$ and $\hat{z}_{ul}$. Finally, in the semi-supervised classification phase, the encoder becomes a classifier, and the latent class variable vector $\hat{y}_l$ is a class score vector. Obviously, we hope that the class score vector $\hat{y}_l$ is as close as possible to the label vector of the sample x_l.

4 The SSAAE-Based Spectrum Sensing Algorithm

In this section, we introduce the SSAAE model into spectrum sensing, and propose an SSAAE-based spectrum sensing algorithm. This algorithm is mainly divided into four modules: data processing, network training, threshold setting and online test. The workflow of each module is as follows:

i.) Data Processing

In general, the input of the SSAAE model is a picture pixel matrix, and then the matrix is expanded into a vector. In this papar our input is directly a vector, and the reconstruction sample is also a vector. Similar to [2], we divide the received signals into real and imaginary parts. So, our input vector is as follows:

$$x = \{\Re(x_1(1)), \Im(x_1(1)), \ldots, \Re(x_1(N)), \Im(x_1(N)), \ldots, \Re(x_M(N)), \Im(x_M(N))\}, \tag{4.1}$$

where $\Re(x_m(n))$ and $\Im(x_m(n))$ denote the real part and the imaginary part of $x_m(n)$, respectively. Therefore, the size of an input x of the SSAAE model is $(2MN, 1)$.

The training set includes two subsets, one subset Ω_l of all the labeled samples and the other subset Ω_{ul} of all the unlabeled samples, which can be expressed as:

$$\begin{aligned} \Omega_l &= \{(x_l^1, y_l^1), (x_l^2, y_l^2), \ldots, (x_l^{K_l}, y_l^{K_l})\}, \\ \Omega_{ul} &= \{x_{ul}^1, x_{ul}^2, \ldots, x_{ul}^{K_{ul}}\}, \end{aligned} \tag{4.2}$$

where K_l and K_{ul} represent the number of labeled samples and unlabeled samples, respectively; $x_l^k (k = 1, 2, \ldots, K_l)$ and $x_{ul}^k (k = 1, 2, \ldots, K_{ul})$ are inputs of the SSAAE model, as described in (4.1). The $y_l^k \in \{[1, 0]^T, [0, 1]^T\} (k = 1, 2, \ldots, K_l)$ are sample labels, where $y_l^k = [1, 0]^T$ and $y_l^k = [0, 1]^T$ represent the H_0 and H_1 hypotheses, respectively.

ii.) Network Training

As described in Section 3, the training of the SSAAE model includes three phases, and the training process of each phase is as follows.

1) Reconstruction phase: Train the encoder and the decoder. Put unlabeled samples $\{x_{ul}^{k_i}\}_{i=1}^m$ into the encoder to obtain $\{\hat{y}_{ul}^{k_i}\}_{i=1}^m$ and $\{\hat{z}_{ul}^{k_i}\}_{i=1}^m$, then reconstructed samples $\{\hat{x}_{ul}^{k_i}\}_{i=1}^m$ are obtained through the decoder, where m is the size of mini-batch and $k_i (i = 1, \ldots, m) \in \{1, \ldots, K_{ul}\}$. The encoder and the decoder are trained using reconstruction loss function:

$$L_R = \frac{1}{m} \sum_{i=1}^{m} \parallel x_{ul}^{k_i} - \hat{x}_{ul}^{k_i} \parallel^2, \tag{4.3}$$

where $\| \cdot \|$ is the Euclidean norm. Minimize L_R to update the encoder and the decoder network parameters θ and φ.

2) Regularization phase: Train the discriminators and the encoder. The encoder plays the role of generator in this phase. First, the categorical discriminator and the Gaussian distribution discriminator are trained. The real samples $\{y_{ul}^{k_i}\}_{i=1}^m$ and the generated samples $\{\hat{y}_{ul}^{k_i}\}_{i=1}^m$ are input into the categorical discriminator D_{ϕ_y}, where ϕ_y represents the categorical discriminator network parameters. And the categorical discriminator is trained using the following loss function:

$$L_{Dy} = -\frac{1}{m} \sum_{i=1}^{m} [log(D_{\phi_y}(y_{ul}^{k_i})) + log(1 - D_{\phi_y}(\hat{y}_{ul}^{k_i}))], \tag{4.4}$$

where $D_{\phi_y}(y_{ul}^{k_i})$ and $D_{\phi_y}(\hat{y}_{ul}^{k_i})$ represent the outputs of the real sample $y_{ul}^{k_i}$ and the generated sample $\hat{y}_{ul}^{k_i}$ in the discriminator D_{ϕ_y} respectively. Minimize L_{Dy} to update ϕ_y.

Similarly, the real samples $\{z_{ul}^{k_i}\}_{i=1}^m$ and the generated samples $\{\hat{z}_{ul}^{k_i}\}_{i=1}^m$ are input into the Gaussian distribution discriminator D_{ϕ_z}, where ϕ_z represents the Gaussian distribution discriminator network parameters. And the Gaussian distribution discriminator is trained using the following loss function:

$$L_{Dz} = -\frac{1}{m} \sum_{i=1}^{m} [log(D_{\phi_z}(z_{ul}^{k_i})) + log(1 - D_{\phi_z}(\hat{z}_{ul}^{k_i}))]. \tag{4.5}$$

where $D_{\phi_z}(z_{ul}^{k_i})$ and $D_{\phi_z}(\hat{z}_{ul}^{k_i})$ represent the outputs of the real sample $z_{ul}^{k_i}$ and the generated sample $\hat{z}_{ul}^{k_i}$ in the discriminator D_{ϕ_z} respectively. Minimize L_{Dz} to update ϕ_z.

Then we update the generator to confuse two discriminators. The generator's loss function L_G can be combined for two adversarial networks:

$$L_G = \frac{1}{m} \big[\sum_{i=1}^{m} log(1 - D_{\phi_y}(\hat{y}_{ul}^{k_i})) + \sum_{i=1}^{m} log(1 - D_{\phi_z}(\hat{z}_{ul}^{k_i})) \big]. \tag{4.6}$$

Minimize L_G to update the encoder network parameters θ.

3) Semi-supervised classification phase: Train the encoder. For labeled samples Ω_l, the encoder acts as a classifier. At this phase, we only use the latent class variable vector $\hat{y}$. The labeled samples $\{x_l^{k_i}\}_{i=1}^m$ are sent to the classifier C_θ, and corresponding outputs $\{\hat{y}_l^{k_i}\}_{i=1}^m$ are class score vectors. We use the cross-entropy loss function to train the classifier network:

$$L_C = -\frac{1}{m}\sum_{i=1}^{m}[\hat{y}_l^{k_i} log(y_l^{k_i}) + (1-\hat{y}_l^{k_i}) log(1-y_l^{k_i})]. \tag{4.7}$$

Minimize L_C to update the encoder network parameters θ.

The above three phases are cyclically trained until the maximum number of iterations is reached. Finally, an optimized classifier C_{θ^*} is obtained, where θ^* is the network parameters of the optimized classifier. The class score vector $\hat{y}$ of a sample x can be expressed as

$$\hat{y} = C_{\theta^*}(x) = \begin{bmatrix} C_{H_0|\theta^*}(x) \\ C_{H_1|\theta^*}(x) \end{bmatrix}, \tag{4.8}$$

where $C_{\theta^*}(x)$ denotes the output of the sample x in the optimized classifier, $C_{H_i|\theta^*}(x)$ represent the class score of H_i $(i = 0, 1)$, and $C_{H_0|\theta^*}(x) + C_{H_1|\theta^*}(x) = 1$.

iii.) Threshold Setting

In general, $C_{H_1|\theta^*}(x) \leqslant 0.5$ signifies the predicted result is H_0, while $C_{H_1|\theta^*}(x) > 0.5$ signifies the predicted result is H_1. However, in order to control the false alarm probability P_f and the detection probability P_d, we need to set a decision threshold γ, namely:

$$\begin{aligned} H_0 &: C_{H_1|\theta^*}(x) \leqslant \gamma, \\ H_1 &: C_{H_1|\theta^*}(x) > \gamma. \end{aligned} \tag{4.9}$$

We put samples of H_0 labeled into the classifier C_{θ^*} to get the class score vectors, and then use Monte Carlo method to calculate the threshold γ corresponding to a fixed false alarm probability P_f.

iv.) Online Test

In the testing phase, we need only the classifier C_{θ^*}. The sampled signals $\{x_m(n)\}$ are converted into the input x_{test} according to the equation (4.1). Then we input x_{test} into the classifier C_{θ^*} to get the class score vector $C_{\theta^*}(x_{test})$. And by comparing $C_{H_1|\theta^*}(x_{test})$ with the set threshold γ, we can get a final classification result.

5 Simulation Results

In our numerical experiments, we use the orthogonal frequency division multiplexing (OFDM) signals modulated by the binary phase shift keying (BPSK) as the PU signals. The OFDM signals are the dominant signals in communications because, they make use of the orthogonality between subcarriers and can send a large amount of data even in a narrow bandwidth. The key parameters are set as follows: the sampling frequency and the carrier frequency are 300 MHz and 100 MHz, respectively; the propagation channel is a frequency selective Rayleigh fading channel and the maximum carrier frequency offset is 250. The SNR is defined as follows:

$$SNR = \frac{E(\parallel s_m(n) \parallel^2)}{E(\parallel v_m(n) \parallel^2)}. \tag{5.1}$$

We assume that the variances of the signals and the noises are σ_s^2 and σ_v^2 respectively. According to the formula (5.1), we have $SNR = \sigma_s^2/\sigma_v^2$. In simulations, the number M of

detection antennas of the SU is set to 4, and in each sensing period the SU samples 100 times ($N = 100$). Then the input size of the SSAAE model is 800×1.

In implementation, we use tensorflow to build and train the SSAAE model. The numbers of labeled samples and unlabeled samples are $K_l = 2 \times 10^3$ and $K_{ul} = 4 \times 10^4$, respectively. In the labeled samples, half of them are "the PU is present" samples, and the other half are "the PU is absent" samples, and the same is true for the unlabeled samples. The setting of the network hyperparameters are shown in Table 5.1. In Table 5.1, lr1 refers to the learning rate of the encoder and the decoder, lr2 refers to the learning rate of two discriminators, β represents the exponential decay rate of first-order moment estimation, and the number of epochs represents the maximum number of iterations. For testing the detection performance of our proposed algorithm, we put 10^4 labeled samples into the well-trained SSAAE model, one half belongs to "the PU is absent", and the other half is of "the PU is present". In order to determine the threshold γ corresponding to any given false alarm probability P_f, we arrange all the output values $C_{H_1|\theta^*}(x_i)$ $(i = k_1, \ldots, k_{5\times10^3})$ corresponding to "the PU is absent" samples in descending order and write them as a vector v. Then we set the threshold γ with the P_f as follows:

$$\gamma = v(\lceil 5 \times 10^3 P_f \rceil), \tag{5.2}$$

where $\lceil \cdot \rceil$ represents the round up to the nearest integer, and $v(i)$ denotes the $i-th$ component of v. Anyway, the above threshold γ is reasonable at least for the 10^4 test samples because, by the formula (4.9), the false alarm rate of these samples under the given threshold (5.2) is as follows:

$$\frac{\text{Cardinality of the set } \{i \mid v(i) > \gamma\}}{5 \times 10^3} \leq \frac{\lceil 5 \times 10^3 P_f \rceil}{5 \times 10^3} \leq P_f. \tag{5.3}$$

Table 5.1: Network Hyperparameter

Hyperparameter	Value
Learning rate	lr1=0.002; lr2=0.001;β=0.9
Batch size	100
Number of epochs	100
Optimizer	Adam

According to the IEEE802.22 standard (the P_d of the SU should be greater than 0.9 and the P_f should not exceed 0.1 [14]), we choose $P_f = 0.1$ and $P_f = 0.01$ respectively, and then compare the performance of the SSAAE-based method with the performances of other methods (ED [4], MED [22], CAV [23], ANN-based method [17] and CM-CNN-based method [9]). ED, MED and CAV are three classical spectrum sensing methods, and ANN-based method and CM-CNN-based method are deep learning-based methods, which all have been introduced in Section 1. For the CM-CNN-based method, due to the small number of detection antennas in our experimental scenario, we adopt the definition method of the covariance matrix in [22], in which the smoothing factor L is set to 7. Both deep learning-based methods use 2×10^3 labeled samples for training, and their experimental results are shown in Fig. 5.2 and Fig. 5.3. From Fig. 5.2 and Fig. 5.3, it can be seen that the detection results obtained by the deep learning-based methods are better than the results obtained by the classical detection methods. And among the given deep learning-based methods, the SSAAE-based method is the best. For example, when $P_f = 0.1$ and SNR=-15dB, the P_d of the SSAAE-based method, CM-CNN-based method, ANN-based method, ED, MED and CAV are 0.832, 0.5696, 0.2634, 0.255 and 0.1528, respectively.

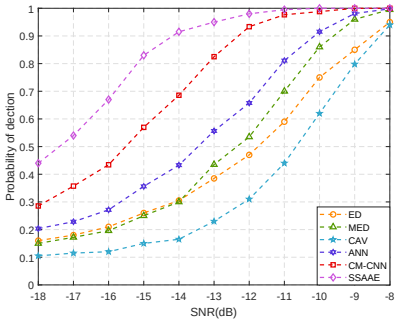

Figure 5.2: P_d vs SNR curves of different algorithms, M=4, N=100, $P_f = 0.1$.

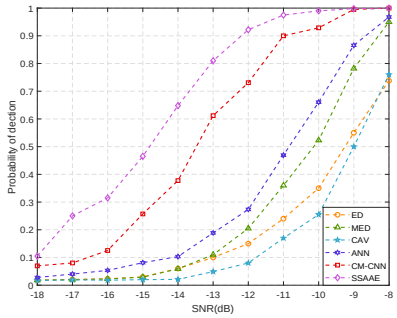

Figure 5.3: P_d vs SNR curves of different algorithms, M=4, N=100, $P_f = 0.01$.

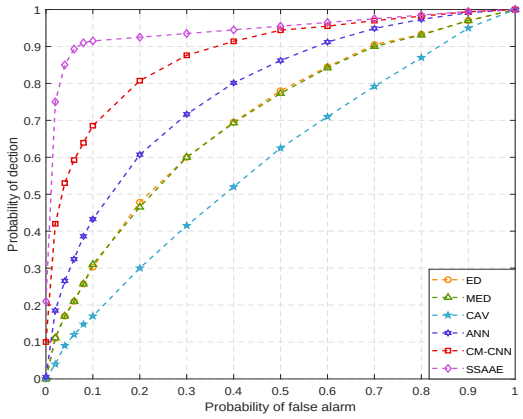

Figure 5.4: ROC curves of different methods, M=4, N=100, SNR=−14dB.

In addition, receiver operating characteristics (ROC) curves are shown in Fig. 5.4 for all the six methods at SNR=−14dB. It can be seen that our proposed method performs significantly better than other methods at low P_f. When the P_f increases from 0 to 0.1, the P_d of the SSAAE-based method increases rapidly from 0.214 to 0.9216, while the P_d of other methods grow slowly.

In order to observe the effectiveness of the unlabeled samples, we train the SSAAE model with zero unlabeled samples, whose experimental results are shown in Fig. 5.5. Obviously, unlabeled samples play an important role in our method performance. For example, when $P_f = 0.1$ and SNR=−15dB, the P_d of the SSAAE-based method can reach 0.832, while the P_d of SSAAE-based method using only the labeled samples can only reach 0.638. By the way, Fig. 5.5, combined with Fig. 5.2, shows that even if we do not use unlabeled samples, our detection results are comparable to that of the CM-CNN-based method, which was the best among all supervised methods.

In order to explore the influence of the number of antennas and sampled signals on the performance of the SSAAE-based spectrum sensing method, we do numerical experiments under different SNR (set $P_f = 0.1$) and the results are shown in Fig. 5.6. When the number

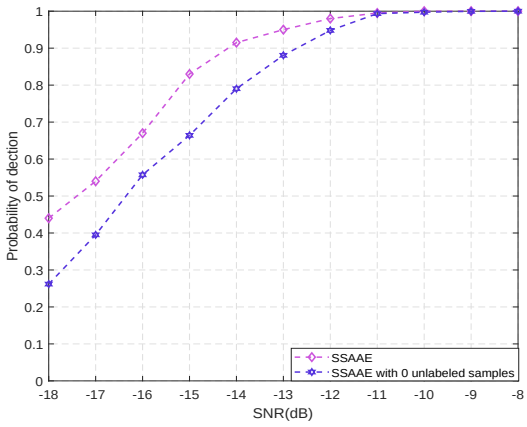

Figure 5.5: P_d vs SNR curves of different input samples, M=4, N=100, $P_f = 0.1$.

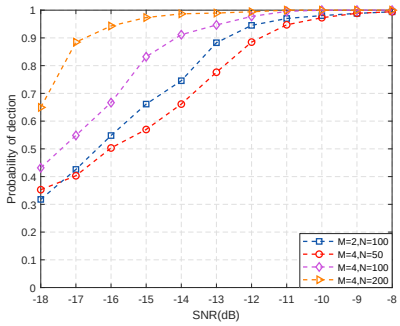

Figure 5.6: P_d vs SNR curves with different numbers of M and N, $P_f = 0.1$.

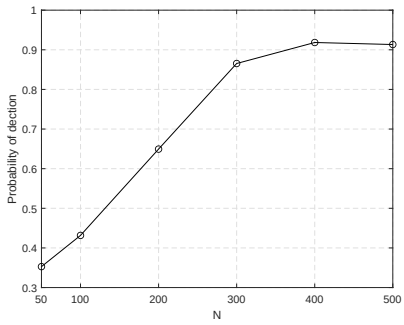

Figure 5.7: Detection probability curve under different N, SNR=−18dB, M=4, $P_f = 0.1$.

of sampled signals N is fixed, it is clearly observed that $M = 4$ performs better than $M = 2$. On the other hand, we can see that as the number of sampled signals N increases, the P_d also increases, but it doesn't go up all the time. As shown in Fig. 5.7, in the case of SNR=−18dB, the P_d of the SSAAE-based method increases rapidly when N increases from 50 to 300, and then the P_d improves slowly until it reaches the upper limit at $N = 400$. The larger the number of sampled signals, the longer the corresponding sensing time, which means that the time for the SU to access target channel for transmission will decrease. So in actual application, the size of N requires a certain trade-off. For example, when the P_d has a minimum requirement, on the premise of meeting this requirement, N should be taken as small as possible to reduce the sensing time.

6 Conclusion

In this paper, an SSAAE-based spectrum sensing algorithm is proposed to detect the activity states of the PU, which is a semi-supervised classification algorithm. Different from supervised spectrum sensing algorithms, the SSAAE-based algorithm requires only a small

amount of labeled samples, which greatly reduces the difficulty of collecting training samples. Moreover, it is a blind detection method, that is, it does not require any prior information about the signals and the noises. At the end of this paper, OFDM signals are used to do our simulation experiments under white Gaussian noises. The simulation results show that the SSAAE-based method is significantly better than classical and supervised deep learning-based spectrum sensing methods under low SNR.

References

[1] E. Axell, G. Leus, E. Larsson and H. Poor, Spectrum sensing for cognitive radio: State-of-the-art and recent advances, *IEEE Signal Processing Magazine* 29 (2012) 101–116.

[2] Q. Cheng, H. Shi, D. Nguyen and E. Dutkiewicz, Sensing OFDM signal: A deep learning approach, *IEEE Transactions on Communications* 67 (2019) 7785–7798.

[3] D. Ciregan, U. Meier and J. Schmidhuber, Multi-column deep neural networks for image classification, in: *2012 IEEE conference on computer vision and pattern recognition*, 2012, pp. 3642–3649.

[4] F. Digham, M. Alouini and M. Simon, On the energy detection of unknown signals over fading channels, in: *IEEE International Conference on Communications. ICC'03*, vol. 5, 2003, pp. 3575–3579.

[5] X. Glorot, A. Bordes and Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: *International Conference on Machine Learning*. 2018.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, Generative adversarial nets, in: *In Advances in neural information processing systems*, 2020, pp. 2672–2680.

[7] D. Kingma and M. Welling, Auto-encoding variational bayes, *arXiv preprint arXiv:1312.6114*. 2020.

[8] C. Liu, X. Liu and Y. Liang, Deep CNN for Spectrum Sensing in Cognitive Radio, in: *2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[9] C. Liu, J. Wang, X. Liu and Y. Liang, Deep CM-CNN for spectrum sensing in cognitive radio, *IEEE Journal on Selected Areas in Communications* 37 (2019) 2306–2321.

[10] J. Lunden, V. Koivunen and H. Poor, Spectrum exploration and exploitation for cognitive radio: Recent advances, *IEEE Signal Processing Magazine* 32 (2015) 123–140.

[11] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow and B. Frey, Adversarial autoencoders, *arXiv preprint arXiv:1511.05644*. 2020.

[12] S. Mittal, Semi-supervised Learning for Real-world Object Recognition using Adversarial Autoencoders, in: *IEEE Transactions on Vehicular Technology*, 2017.

[13] J. Mitola and G. Maguire, Cognitive radio: Making software radios more personal, *IEEE Personal Communications* 6 (1999) 13–18.

[14] S. Shellhammer, Spectrum sensing requirements summary, *Doc. IEEE 802.22-06/0089r4*. 2006.

[15] R. Tandra and A. Sahai, SNR walls for signal detection, *IEEE Journal of selected topics in Signal Processing* 2 (2008) 4–17.

[16] Y. Tang, Q. Zhang and W. Lin, Artificial neural network based spectrum sensing method for cognitive radio, in: *In 2010 6th International Conference on Wireless Communications Networking and Mobile Computing*, 2010, pp. 1–4.

[17] M. Vyas, D. Patel and M. Lopez-Benitez, Artificial neural network based hybrid spectrum sensing scheme for cognitive radio, in: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, 2017, pp. 1–7.

[18] J. Xie, J. Fang, C. Liu and Y. Yang, Unsupervised Deep Spectrum Sensing: A Variational Auto-Encoder Based Approach, *IEEE Transactions on Vehicular Technology* 69 (2020) 5307–5319.

[19] J. Xie, C. Liu, Y. Liang and J. Fang, Activity pattern aware spectrum sensing: A CNN-based deep learning approach, *IEEE Communications Letters* 23 (2019) 1025–1028.

[20] S. Xu, Z. Zhao and J. Shang, Spectrum sensing based on cyclostationarity, in: *2008 Workshop on Power Electronics and Intelligent Transportation System*, 2008, pp. 171–174.

[21] T. Young, D. Hazarika, S. Poria and E. Cambria, Recent trends in deep learning based natural language processing, *IEEE Computational intelligence Magazine* 13 (2018) 55–75.

[22] Y. Zeng, C. Koh and Y. Liang, Maximum eigenvalue detection: Theory and application, in: *2008 IEEE International Conference on Communications*, 2008, pp. 4160–4164.

[23] Y. Zeng and Y. Liang, Spectrum-sensing algorithms for cognitive radio based on statistical covariances, *IEEE transactions on Vehicular Technology* 58 (2008) 1804–1815.

[24] Z. Zivkovic, Improved adaptive Gaussian mixture model for background subtraction, *Proceedings of the 17th International Conference on Pattern Recognition* 2 (2004) 28–31.

Manuscript received 19 July 2021
revised 18 October 2021
accepted for publication 25 November 2021

Wei Liang
School of Science, Beijing University of Posts and Telecommunications
Beijing, 100876, China
E-mail address: E-mail: liangwei@bupt.edu.cn

Xi Zhang
School of Science, Beijing University of Posts and Telecommunications
Beijing, 100876, China
E-mail address: zhang_xi@bupt.edu.cn

Jianhua Yuan
School of Science, Beijing University of Posts and Telecommunications
Beijing, 100876, China
E-mail address: E-mail: jianhuayuan@bupt.edu.cn

Caixia Kou
School of Science, Beijing University of Posts and Telecommunications
Beijing, 100876, China
E-mail address: E-mail: koucx@bupt.edu.cn

Wenbao Ai
School of Science, Beijing University of Posts and Telecommunications
Beijing, 100876, China
E-mail address: E-mail: aiwb@bupt.edu.cn