IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 10, 2019, accepted March 28, 2019, date of publication April 11, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909976

Exploiting Sparsity Recovery for Compressive Spectrum Sensing: A Machine Learning Approach

MAHMOUD NAZZAL[1], (Member, IEEE), ALİ RIZA EKTİ[2,4], (Member, IEEE), ALİ GÖRÇİN[3,4], (Member, IEEE), AND HÜSEYİN ARSLAN[1,5], (Fellow, IEEE)

[1]School of Engineering and Natural Sciences, Istanbul Medipol University, 34810 Istanbul, Turkey
[2]Department of Electrical and Electronics Engineering, Balıkesir University, 10100 Balıkesir, Turkey
[3]Faculty of Electronics and Communications Engineering, Yıldız Technical University, 34220 Istanbul, Turkey
[4]Informatics and Information Security Research Center (BİLGEM), TÜBİTAK, 41470 Gebze, Turkey
[5]Department of Electrical Engineering, University of South Florida, Tampa, FL 33620, USA

Corresponding author: Mahmoud Nazzal (mahmoud.nazzal@ieee.org)

This work was supported by the Scientific and Research Council of Turkey (TUBITAK).

ABSTRACT Sub-Nyquist sampling for spectrum sensing has the advantages of reducing the sampling and computational complexity burdens. However, determining the sparsity of the underlying spectrum is still a challenging issue for this approach. Along this line, this paper proposes an algorithm for narrowband spectrum sensing based on tracking the convergence patterns in sparse coding of compressed received signals. First, a compressed version of a received signal at the location of interest is obtained according to the principle of compressive sensing. Then, the signal is reconstructed via sparse recovery over a learned dictionary. While performing sparse recovery, we calculate the sparse coding convergence rate in terms of the decay rate of the energy of residual vectors. Such a decay rate is conveniently quantified in terms of the gradient operator. This means that while compressive sensing allows for sub-Nyquist sampling thereby reducing the analog-to-digital conversion overhead, the sparse recovery process could be effectively exploited to reveal spectrum occupancy. Furthermore, as an extension to this approach, we consider feeding the energy decay gradient vectors as features for a machine learning-based classification process. This classification further enhances the performance of the proposed algorithm. The proposed algorithm is shown to have excellent performances in terms of the probability-of-detection and false-alarm-rate measures. This result is validated through numerical experiments conducted over synthetic data as well as real-life measurements of received signals. Moreover, we show that the proposed algorithm has a tractable computational complexity, allowing for real-time operation.

INDEX TERMS Sparse coding, spectrum sensing, machine learning classification, residual components.

I. INTRODUCTION

The increasing demand on high data rate communications highlights the scarcity of available spectrum [1]. Standard spectrum usage is based on allocating a specific frequency resource to each user. However, this policy is widely believed to under-utilize the wireless spectrum [2]. Accordingly, this calls for exploiting the unused spectrum portions, referred to as spectral opportunities for communication. To detect spectral opportunities, reliable and fast spectrum sensing (SS) is required [3].

SS approaches can be mainly classified into narrowband methods and wideband methods. In this context, the term *narrowband* means that the frequency range is sufficiently narrow such that the channel frequency response can be considered as flat. Standard narrowband SS techniques can be classified into three broad categories; matched filtering [4], energy detection [5], and cyclostationary feature detection [6]. This work considers the narrowband scenario of SS.

Matched filtering SS archives optimum detection by correlating the received signal with a pre-defined signal pattern. Despite its optimality, this method requires prior knowledge about the signal transmitted by the primary users (PU)s. On the contrary, energy detection avoids the need for prior knowledge of the PUs at lower implementation and computational complexity levels. Still, it performs poorly in low SNR scenarios. Cyclostationary feature detection distinguishes

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

between different types of primary signals by exploiting their cyclostationary features. However, the computational cost of such an approach is relatively high.

Recent SS literature considers using compressive sensing (CS) as a framework for SS [7]. Similar to the witnessed success of CS in various application areas, the fundamental gain CS offers is alleviating the need for high-sampling-rate analog-to-digital converters (ADC)s and costly radio frequency (RF) circuitry [8].

A. MOTIVATION AND RELATED WORKS

CS-based SS considers sampling the spectrum with a few measurements at a rate often lower than the Nyquist rate. Hence, this SS scheme is referred to as the sub-Nyquist SS. This leads to reduced energy consumption, complexity, and memory requirements [9]. Sub-Nyquist measurements are then used to approximately reconstruct the spectrum.

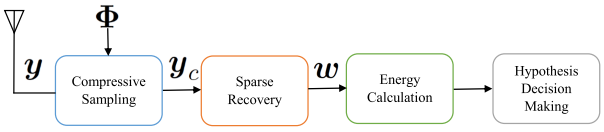


FIGURE 1. Compressive spectrum sensing in block diagram.

The main rationale of the above-explained CS-based SS approach is depicted in Fig. 1. The spectrum $\boldsymbol{y}$ is down-sampled as $\boldsymbol{y}_c$ using a CS matrix. Then, the aim is to reconstruct $\boldsymbol{y}$ from $\boldsymbol{y}_c$ in a certain domain. It is customary to expand $\boldsymbol{y}_c$ in the discrete Fourier transform (DFT) domain to as $\hat{\boldsymbol{y}}_c = \boldsymbol{F}_n \boldsymbol{w}$. From the DFT coefficients, there are various approaches to SS. For example, spectral domain energy detection [5] is a typical approach. This is based on calculating the energy of the reconstructed signal and comparing it with a specific energy threshold to perform hypothesis testing. Other works [10] consider exploiting the statistical properties of the reconstructed signals for the purpose of SS.

The sub-Nyquist SS paradigm is based on assuming spectrum sparsity with respect to some domain or transformation [11]. Fortunately, this is true for under-utilized spectra in the frequency domain [12], as the number of PUs is smaller than what the spectrum can accommodate for. Moreover, the spectrum can exhibit a sparse nature in other domains, especially, in the case of overcomplete learned dictionaries.

Knowing the exact sparsity level is of crucial importance to the success of sub-Nyquist SS. In the signal acquisition stage, the assumed sparsity governs the number of sub-Nyquist samples. More specifically, the minimal number of samples to guarantee an efficient recovery is $M = C_0\, S \log(M_{Nyq}/S)$ [13] where C_0 is a constant, M_{Nyq} the number of measurements at the Nyquist rate, and S the assumed sparsity. In the recovery stage, the sparsity level determines the quality of the reconstructed spectra.

To reconstruct the spectrum from its sub-Nyquist measurements, sparse recovery can be applied in one of the following two formulations. First, is to minimize the reconstruction error with a fixed sparsity level [14]. Clearly, this approach requires prior knowledge about the sparsity level. However, the sparsity level is dependent on spectrum occupancy [15], which is usually unavailable to either the dynamic activities of PUs or the time-varying fading channels [12]. Secondly, is to minimize sparsity level such that the reconstruction error is upper-bounded with a certain error tolerance level [16]. This requires information about the noise level and signal-to-noise ratio values of the PU signals, which should not either be available. Based on either of these two approaches, other more advanced sparse recovery techniques have been proposed. For example, [17] adopts minimizing trigonometric functions as a means of a better approximating the l_0 norm.

Along the line of sparsity level-based reconstruction, [18] proposes a denoised version of the CS framework for SS to suppress the impact of inherent noise. However, this approach assumes knowing the sparsity level to minimize the sampling rate. This requirement is alleviated in [19] where one first estimates the sparsity level, and then applies CS accordingly. Nevertheless, this sparsity level estimation is computationally expensive.

Another attempt at avoiding sparsity level estimation is proposed by Sun *et al.* [20]. This is done by iteratively performing step-by-step CS processes while setting the number of measurements in an adaptive manner. However, this iterative process incurs higher computational complexities as sparse coding has to be solved several times until the exact signal recovery is achieved. As a more efficient alternative, Qin *et al.* [21] covey an approximate sparsity level from geolocation data. This low-complexity estimation allows for efficient performance at relatively lower computational complexity levels.

B. CONTRIBUTIONS, ORGANIZATION AND NOTATION

This paper proposes an algorithm for narrowband sub-Nyquist SS based on tracking the convergence in energy decay of sparsely coded compressed received signals. Here is a summary of the contributions presented in this paper.

- This approach alleviates the need for estimating the sparsity level, neither a specific sparse coding error tolerance. This is because sparse recovery is carried out for revealing its energy convergence rate, rather than giving an accurate signal reconstruction. Hence, if the actual sparsity level is exceeded, that will not affect the convergence rate, and therefore the decision to be made. Besides, the sparsity level used is independent of the measurements undertaken in the sensing stage.
- This work comes along the line of CS-based methods. Sub-Nyquist sampling reduces the analog-to-digital conversion overhead. The proposed algorithm uses learned dictionaries instead of fixed basis functions, thereby allowing for better sparse recovery, improved noise rejection and lower numbers of samples for efficient signal acquisition.
- This work is also applicable with sampled dictionaries as a means of substantially reducing the computational cost, without sacrificing the performance.

- The absolute gradient is applied as a feature vector fed to a machine learning (binary) classifier to enhance the classification quality. Such a classifier can be a support-vector machine classifier (SVM) or a deep neural network classifier (DNN).

The proposed algorithm is shown to efficiently identify spectral opportunities in terms of the probability-of-detection and false alarm rate measures. This result is validated through performance analysis tests conducted over synthetic data as well as real-life measurements of received signals. We show that the proposed algorithm has a tractable computational complexity, and can, therefore, be implemented in real-time.

The remainder of the paper is organized as follows: In Section II, a brief background of the main topics is provided. Section III presents the proposed algorithm. Synthetic and measurement-based experiments are provided in Section IV, with the conclusions made in Section V.

Notation: Plain-faced letters represent scalars. Bold-faced lower-case, and bold-faced upper-case letters denote vectors and matrices, respectively. In a matrix $\boldsymbol{X}$, the symbol $\boldsymbol{X}_i$ resembles the i-th column in this matrix. Similarly, $\boldsymbol{x}_i$ is the i-th element in the vector $\boldsymbol{x}$. The $\|.\|_2$, $\|.\|_0$ and $\langle\cdot,\cdot\rangle$ symbols represent the 2-norm, the number of nonzero elements in a vector, and the inner product operator, respectively.

II. BACKGROUND

A. COMPRESSIVE SENSING

The fundamental assumption behind CS is that the majority of a signal's salient information content is contained in a relatively small number of random signal projections [22]. This allows for sub-Nyquist sampling of signals without a noticeable loss in information if one wisely selects where and when to sense [2]. Such a reduced sampling rate results in several advantages of; reducing the circuit and time cost of sensing, and allowing for efficient low-dimensional representation and processing of signals.

Real life signals are either sparse or compressible. Sparse signals have a few nonzero elements, as they are; in the standard basis. However, compressible ones are sparse in a certain domain/transformation. The above mentioned CS paradigm is applicable for both sparse and compressible signals. Hence, it is generally applicable to natural signals.

If $\boldsymbol{y} \in \mathbb{C}^N$ is a sparse signal, one can obtain a corresponding compressed version $\boldsymbol{y}_c \in \mathbb{C}^M$, using a sensing matrix $\boldsymbol{\Phi} \in \mathbb{C}^{M\times N}$, as follows.

$$\boldsymbol{y}_c = \boldsymbol{\Phi}\boldsymbol{y}, \tag{1}$$

where M is the number of CS measurements. Herein, $\boldsymbol{\Phi}$ denotes a sensing matrix that selectively chooses the signal samples to be sensed. Typically, there are two categories of sensing matrices; deterministic and random. It is shown that random matrices such as Gaussian or Bernoulli perform very well in practice.

From a low dimensional measurement, a high-dimensional signal version can be reconstructed. This is often achieved through a sparse recovery process.

B. SPARSE RECOVERY AND DICTIONARY LEARNING

If a signal $\boldsymbol{y} \in \mathbb{C}^N$ admits sparse coding over a dictionary $\boldsymbol{D} \in \mathbb{C}^{N\times K}$, then $\boldsymbol{y} \approx \boldsymbol{D}\boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{C}^K$ is said to be a sparse coding coefficient vector. For a given $\boldsymbol{y}$ and $\boldsymbol{D}$, $\boldsymbol{w}$ can be obtained through the following sparse coding process.

$$\underset{\boldsymbol{w}}{arg\,min}\ \|\boldsymbol{w}\|_0\ s.t.\ \|\boldsymbol{y}-\boldsymbol{D}\boldsymbol{w}\|_2^2 < \epsilon, \tag{2}$$

where ϵ is the error tolerance.

The above-mentioned problem is N-P hard. However, there are methods to obtain efficient approximate solutions. These can be classified into two main categories. First, is l_1 relaxation methods, where one relaxes the l_0 measure to the l_1, thereby offering a significant reduction to the computational complexity of the problem. An example of these methods is the basis pursuit algorithm and its variants, such as [16]. The other category is greedy pursuit algorithms that solve the problem iteratively, such as the matching pursuit methods [14]. Algorithms in this category possess the advantages of low computational complexity, but the solution is not necessarily the sparsest.

A collection of predefined basis functions such as the DFT can be used as a dictionary. However, learning a redundant dictionary over a set of training data points $\boldsymbol{Y} \in \mathbb{C}^{N\times L}$ is a better alternative [23] that assures sparsity, enhances representation quality and is locally-adaptive to the signals of interest. Such a dictionary has the basis vectors as its columns (atoms). This is referred to as the dictionary learning process, formulated as

$$\underset{\boldsymbol{W},\boldsymbol{D}}{arg\,min}\ \|\boldsymbol{W}_i\|_0\ s.t.\ \|\boldsymbol{Y}_i-\boldsymbol{D}\boldsymbol{W}_i\|_2^2 < \epsilon\ \forall\ i. \tag{3}$$

An example widely-used algorithm for dictionary learning is the K-SVD algorithm [23].

C. MACHINE LEARNING FOR CLASSIFICATION

1) SVM CLASSIFICATION

Support vector machines (SVM) [24] is a widely used machine learning algorithm for classification and regression tasks. In binary classification, SVM aims at finding an N-dimensional hyperplane separating the data, assuming it exists. It is customary to apply SVM on some distinctive features of the given data. Such a hyperplane can be found such that the following two criteria are met. First, is maximizing the distance between the hyperplane and each of the data points in each pair of closest points in the two classes. Second, each data point should be labeled to belong in its own class. Formally, these requirements are formulated as follows.

$$\begin{aligned}\underset{\boldsymbol{\omega},b}{arg\,min}\ \boldsymbol{f}(\boldsymbol{\omega}) &= \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\\ s.t.\ y_i(\boldsymbol{\omega}^T\boldsymbol{\gamma}(\boldsymbol{x}_i)+b) &\geq 1-\zeta_i\\ \zeta_i \geq 0,\quad \forall\, i &= 1,\ldots,L,\end{aligned} \tag{4}$$

where $\boldsymbol{\omega}$ is a weight vector, b is a bias parameter, $\{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_L,y_L),\boldsymbol{x}\in\mathbb{R}^N, y\in\{+1,-1\}\}$ is a training set. Herein, each 2-tuple consists of an observation vector

$\boldsymbol{x}_i$, along with its known label y_i. It is assumed that this training set can be separated by the hyperplane satisfying $\boldsymbol{\omega}^T \boldsymbol{\gamma}(\boldsymbol{x}) + b = 0$. Besides, the feature map $\boldsymbol{\gamma}(\boldsymbol{x})$ maps $\boldsymbol{x}$ into a higher dimensional space, C is a positive constant, and ζ_i is the error in the soft margin.

The problem formulation in (4) is referred to as the primal formulation. Aside from the primal, it is customary to consider solving a dual counterpart of this formulation. Solving the dual transforms the problem into a quadratic programming problem with the number of variables equal to the number of training data pairs (L). For a given kernel matrix $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\gamma}(\boldsymbol{x}_i)\boldsymbol{\gamma}(\boldsymbol{x}_j)$, training SVM with the dual formulation consists of solving the following optimization problem.

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{arg\,min}\ & \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L} y_i y_j \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{j=1}^{L} \boldsymbol{\alpha}_j \\ s.t.\ & \sum_{i=1}^{L} y_i \boldsymbol{\alpha}_i = 0 \\ & 0 \le \boldsymbol{\alpha}_i \le C, \quad \forall\, i = 1, \ldots, L. \end{aligned} \tag{5}$$

where $\boldsymbol{\alpha}$ is the argument weight vector, L is the length of the SVM training set, C is a positive constant. Here, there is not dependency on $\boldsymbol{\omega}$ or b, and $\boldsymbol{\alpha}$ is the only argument.

After SVM training, labeling an incoming test signal $\boldsymbol{x}$ can be done with the following decision function.

$$\boldsymbol{f}(\boldsymbol{x}) = \text{sign}\Big(\sum_{i=1}^{L} \boldsymbol{\alpha}_i y_i \boldsymbol{K}(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b\Big). \tag{6}$$

This is, in fact, a quadratic programming problem that is easy to solve by standard linear programming.

2) DNN CLASSIFICATION

DNN is in essence an artificial neural network that applies several hidden layers between the input and the output layers. These layers are composed of neurons. The success of DNNs has been widely witnessed in various application areas [25]. This success relies basically on DNN's ability in extracting high-level features from raw sensory data after using statistical learning over a large amount of data to obtain an effective representation of an input space. This does not require handcrafted features or rules designed by experts. In general, the sigmoid function and the rectified linear unit (ReLU) function are the most universal choices in the nonlinear operation, which can be given by

$$\begin{aligned} f_{Sigmoid}(x) &= \frac{1}{1+e^{-x}}, \\ f_{ReLU}(x) &= \max(0, x), \end{aligned} \tag{7}$$

where x is the argument of the function. The input (x) and output)$\mathbf{o}$) of the DNN are related as follows

$$\mathbf{o} = f(\mathbf{x}, w) = f^{(n-1)}\left(f^{(n-2)}\left(\ldots f^{1}(\mathbf{x})\right)\right), \tag{8}$$

Here, n and w denote the number of layers and the associated weights, respectively.

D. SYSTEM MODEL

A communications system is assumed where at the transmitter side, the transmit signal model can be expressed as follows.

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{9}$$

where $\boldsymbol{s} = [s_1(t), \ldots, s_N(t)]^T$ represents N independent data points. Also, $s_i(t) = \sum_{k=-\infty}^{\infty} b_k u(t - kT_s) e^{j2\pi f_{c,n} t}$, where $n = 1, 2, \ldots, N$, and $u(t)$ is the pulse shaping filter, b_k are the digitally modulated symbols, $f_{c,n}$ is the center frequency, and T_s is the symbol duration. $\boldsymbol{A}$ is $n \times n$ coefficient matrix with $a_{i,j}$ elements where $i, j = 1, \ldots N$.

The signal $\boldsymbol{x}$ is then transmitted over a noise channel $\boldsymbol{H}$. The signal received at the receiver side is as follows.

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{10}$$

where $\boldsymbol{y}$ is the received signal, $\boldsymbol{H}$ is the channel matrix (or vector), $\boldsymbol{x}$ is the transmitted signal, and $\boldsymbol{n}$ is additive white Gaussian noise.

This work aims at estimating the spectrum in the above-explained communication system. This sensing is characterized by identifying spectral opportunities. This can be achieved by detecting transceiver communication activity at a given location. However, it is noted that detecting transceivers is a challenging process in practice, as many PUs are passive receivers. Therefore, it is more feasible to detect the existence of signals coming from PUs at a given location [12]. In accordance with the common practice in the SS literature, this is done by deciding whether there is a transmitted signal within the received one, signifying the existence of a PU at that location of interest, and vice-versa. Formally, we target at distinguishing between the following two hypotheses:

- Null Hypothesis ($\mathcal{H}_0$): There is no active PU at the location of interest. Hence, the received signal is merely noise $\boldsymbol{y} = \boldsymbol{n}$.
- Alternate Hypothesis ($\mathcal{H}_1$): A PU exists. Therefore, the signal received at the user location is composed of the transmitted PU signal along with noise. $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$.

The distinction between these two hypotheses should be incoherent, i.e., assumes no prior knowledge about the signaling or modulation of the PU, neither a collaboration from its side.

III. SPARSE CODING RESIDUAL ENERGY DECAY SPEED TRACKING FOR SPECTRUM SENSING

A. SPARSE CODING CONVERGENCE AS MEANS OF TRANSMITTED SIGNAL IDENTIFICATION

As opposed to noise, natural signals are generally compressible. Accordingly, the magnitudes of their sparse coding coefficients are shown to obey a power-law decay [26]. This means that a signal is written in terms of its sparse approximation as $\boldsymbol{x} \approx \boldsymbol{D}\boldsymbol{w}$, and the coefficients $\boldsymbol{w}_i$ are sorted such that $|\boldsymbol{w}_1| \ge |\boldsymbol{w}_2| \ldots \ge |\boldsymbol{w}_n|$, then there exist C and q such that:

$$|\boldsymbol{w}_i| \le C i^{-q}. \tag{11}$$

The larger q is, the faster the magnitudes decay, and the more compressible a signal is.

In general, sparse coding methods such as the greedy OMP [14] iteratively calculate the sparse coding coefficient vector $\boldsymbol{w}$ with a specified sparsity S as follows. First, a so-called residual vector $\boldsymbol{r}_i$, $\{i = 0, 1, \ldots, S-1\}$ is initialized with the signal itself as $\boldsymbol{r}_0 = \boldsymbol{x}$. In each iteration, sparse coding selects one atom from a given dictionary $\boldsymbol{D}$ that best approximates the current residual $\boldsymbol{r}_i$. The residual after each atom selection is updated by subtracting the projection of the residual onto the selected atom from the residual itself. This process is repeated S times.

To this end, let us think of the signal as a composition of its residual components, i.e., $\boldsymbol{x} \approx \sum_{i=1}^{S} \boldsymbol{r}_i$. Each residual component corresponds to one nonzero entry in $\boldsymbol{w}$. In view of the power-law decay property of the elements in $\boldsymbol{w}$, it is evident that the energy of the residual components does also follow the same power-law decay. Stated in the reverse direction, compliance of the energy of the residual components of a signal signifies its compressibility. Known that signal is compressible assures that it is not only noise, but it also contains a transmitted signal component. This relies on the key assumption that noise is not structured, and therefore not compressible. This observation suggests that one can depend on this so-called residual energy decay speed to identify the existence of a PU at a given location.

In harmony with the concept of signal compressibility, it is intuitively expected that the residual vector of a compressible signal exhibits a significant loss in its energy in the first few sparse coding iterations. This is due to the fact that the signal is compressible in the given dictionary, meaning that there is a similarity between such a signal and the dictionary atoms to guarantee a sparse and accurate representation. This is because dictionary atoms are, in fact, prototype signal signals conveyed from real-life training data sets. Stating this in the reverse direction, a fast decay in the residual energy signifies the existence of a compressible signal.

Modulated signals are generally compressible. Hence, this informs on the existence of transmitted signals. On the contrary, the absence of a PU means that the received signal is mere noise. Therefore, the decay speed of the residual will not necessarily be that fast. Besides, it can exhibit chaotic patterns owing to the randomness of noise, and the fact that no single dictionary atom is well-suited to represent noise. Furthermore, this observation can be exaggerated if we consider performing a full atom selection, i.e., selecting all the atoms in the dictionary and calculating the corresponding coefficients. Under this condition, the aforementioned observation will be, especially, more strongly valid. An analysis of this idea is provided in Appendix.

The gradient operator quantifies the speed of decay. For a vector signal $\boldsymbol{x}$, the gradient is the first-order derivative. The gradient of a discrete signal $\boldsymbol{x}$ at time instant τ, it is defined as follows.

$$\boldsymbol{G}(\tau) = \boldsymbol{x}(\tau + 1) - \boldsymbol{x}(\tau). \tag{12}$$

To this end, the gradient operator can guide on how fast the residual energy decays versus iteration. The following test empirically motivates the usage of this operator to guide on the existence of transmitted signals.

B. A MOTIVATING EXAMPLE

In this test, we use the dictionary and setup used in Section IV. We also generate a test set of 10^4 received signals. However, each received signal realization is generated according to both $\mathcal{H}_0$ and $\mathcal{H}_1$. Using a compression ratio of 30%, a compressed version of each test received signal is obtained according to (13). To this end, we perform a sparse coding process of the compressed received signal over $\boldsymbol{D}$, using the OMP [14] algorithm. After each OMP iteration, we calculate the energy of the residual $\|\boldsymbol{r}\|_2$. Afterwards, we calculate the absolute values of the gradient $|\boldsymbol{G}|$ of the energy vector. This process is repeated for 10^4 trials, over a set of several SNR values. The results of this experiment are depicted in Fig. 2.

In view of Fig. 2-a, it is seen that $\|\boldsymbol{r}\|_2$ decays very fast in the first few iterations for $\mathcal{H}_1$; the major energy reduction happens in these iterations. Also, the decay speed afterward fluctuates, but it is still not as high as in the first few iterations. On the contrary, the speed of energy decay fluctuates for the case of $\mathcal{H}_0$. It can be seen that the fastest decay does not happen in the first few iterations. Observing Fig.s 2-b, Fig.s 2-c and 2-d, the same observation can be made.

This test suggests that one can rely on the profile of this gradient versus iteration as a feature vector to signify the underlying hypothesis.

C. THE PROPOSED ALGORITHM

Based on the above motivation, we propose an algorithm for SS through exploiting the residual energy decay with the following two variants.

1) USING THE GRADIENT OPERATOR DIRECTLY

In view of the results reported in Fig. 2, the gradient operator can be directly used for hypothesis testing. This requires developing a test statistic for that purpose. An immediate approach is to measure the first few maxima in the absolute gradient vector. If such maxima happen to be in the first few iterations, this signifies the existence of a transmitted signal and hence a PU, and vice-versa. This simplistic hypothesis testing is outlined Algorithm 1 which represents the run-time stage of the algorithm. Prior to this stage, a training stage is required to train for the dictionary.

It is noted that in the case of $\mathcal{H}_1$, the majority of maxima lies in the first half of the vector $|\boldsymbol{G}|$, and vice-versa. To this end, let us denote by $|\boldsymbol{G}_f|$ and $|\boldsymbol{G}_s|$ the first and second halves of this vector. Besides, let us denote by $\max(\boldsymbol{x}, i)$ the i-th order maximum of a given vector $\boldsymbol{x}$. Then, the underlying hypothesis testing can be formally stated as follows.

$$\sum_{i=1}^{z} \max(|\boldsymbol{G}_f|, i) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \sum_{i=1}^{z} \max(|\boldsymbol{G}_s|, i), \tag{13}$$

where z denotes the number of maxima to consider in the comparison.

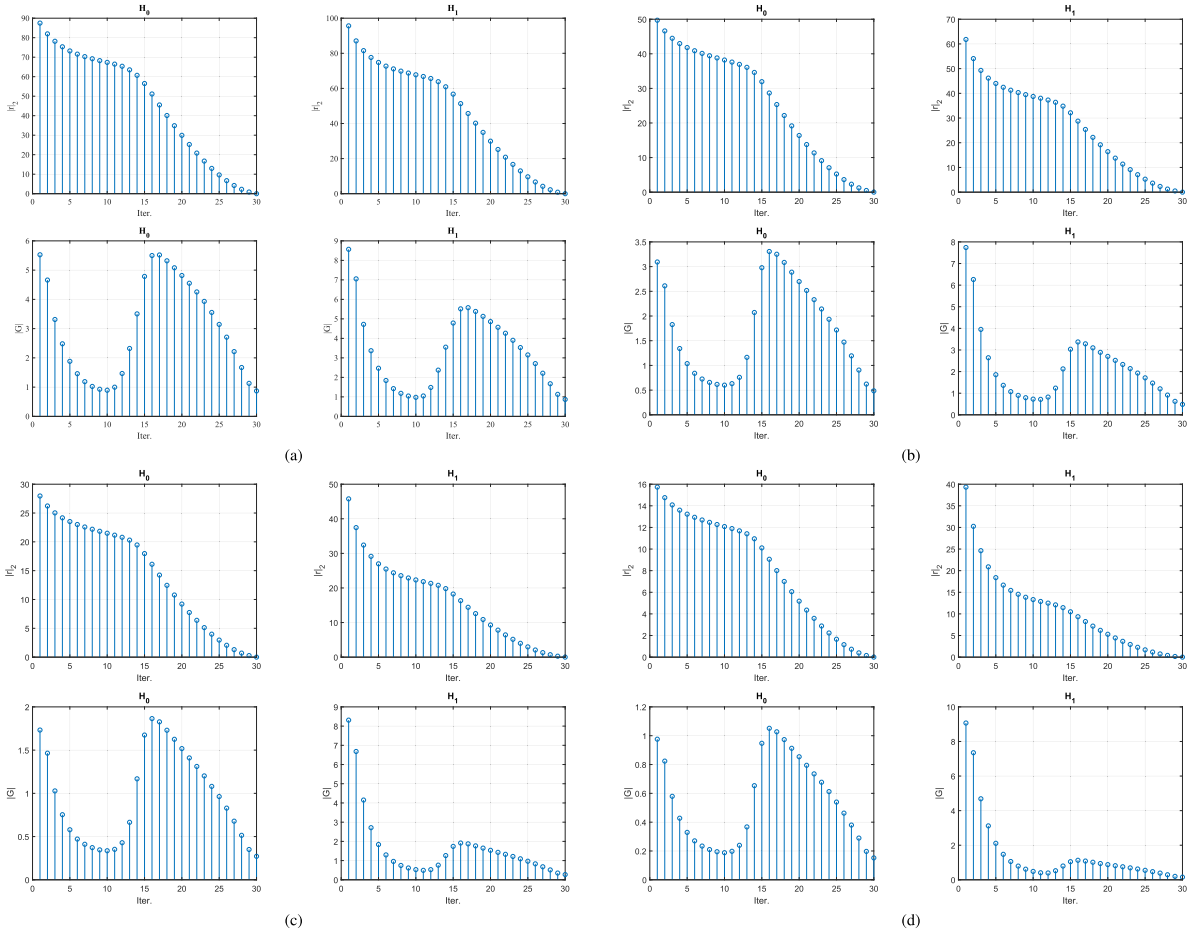


FIGURE 2. The averages of $\|r\|_2$ (up) and $|G|$ (down) versus sparse coding iteration for received signals under $\mathcal{H}_0$ (left) and $\mathcal{H}_1$ (right). In (a), (b), (c), and (d), the SNR values are -5, 0, 5, and 10 *dB*, respectively.

Algorithm 1 PU Sensing via SC Residual Energy Tracing

Input: Dictionary $\boldsymbol{D} \in \mathbb{C}^{N\times K}$, sensing matrix $\boldsymbol{\Phi} \in \mathbb{C}^{M\times N}$, and access to the received signal.

Output: A *decision* about PU existence.

1: Sample a compressed received signal as $\boldsymbol{y}_c = \boldsymbol{\Phi y}$
2: Solve $\underset{\boldsymbol{w}}{arg\,min}\ \|\boldsymbol{y}_c - \boldsymbol{\Phi D w}\|_2^2\ s.t.\ \|\boldsymbol{w}\|_0 = M$
3: Record the energy of the residual for each iteration.
4: Locate the first few maxima of the absolute gradient of the residual
5: **if** maxima happen in the first few iterations **then**
6: *decision* $\leftarrow 1$
7: **else**
8: *decision* $\leftarrow 0$
9: **end if**
10: **return** *decision*

2) USING THE GRADIENT OPERATOR AS FEATURE FOR MACHINE LEARNING-BASED CLASSIFICATION

As a natural enhancement to Algorithm 1, the absolute gradient vector can be regarded as a feature vector for a better classifier. Generally speaking, one can select an advanced classifier such as machine learning-based classification techniques. Typical examples along this line include SVM and DNN classifiers. In this context, the absolute gradient vectors represent the features of the observed data points. For each gradient vector, the corresponding hypotheses $=$ ($\mathcal{H}_0$ or $\mathcal{H}_1$) is its class label. Hence, a classifier model is trained over a set of gradient operator vectors, with known hypotheses. This process is pictorially described in Fig. 3. It is also noted that one can employ other simpler classification techniques at the expense of losing the performance. Examples include simple perceptron, logistic regression, and naive Bayes classifier.

It is noted that classifier training signals can be either synthetically generated or obtained as field measurements. In either case, one needs to know the hypothesis resembling the class index, and the received signal. Hence, received signals are recorded along with their underlying class indices. Then, the absolute gradient vector of each signal is calculated and used as the feature vector over which the classifier is trained.

Once the classification model is trained, it can be used for classifying an upcoming received signal to be in either

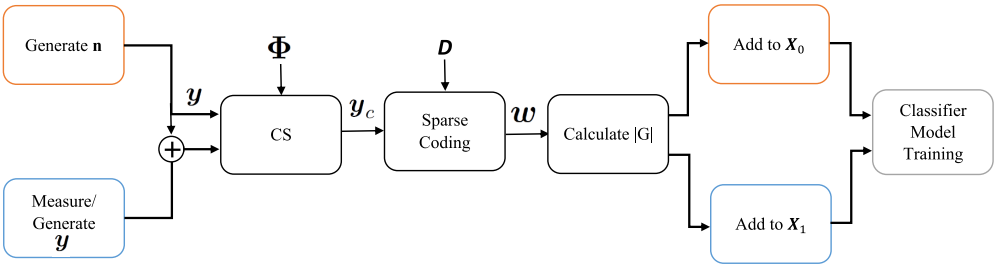


FIGURE 3. Classifier model training in block diagram.

Algorithm 2 PU Sensing via SC Residual Energy Tracing and a ML Classifier

Input: Dictionary $\boldsymbol{D} \in \mathbb{C}^{N\times K}$, sensing matrix $\boldsymbol{\Phi} \in \mathbb{C}^{M\times N}$, an ML classification model, and access to the received signal.

Output: A *decision* about PU existence.

1: Sample a compressed version of the received signal $\boldsymbol{y}_c = \boldsymbol{\Phi y}$
2: Solve $\underset{\boldsymbol{w}}{arg\,min}\ \|\boldsymbol{y}_c - \boldsymbol{\Phi D w}\|_2^2\ s.t.\ \|\boldsymbol{w}\|_0 = M$
3: Record the energy of the residual for each iteration.
4: Calculate the gradient of the residual
5: Using the ML classifier, make a *decision*
6: **return** *decision*

hypothesis based on its gradient vector. The main steps of this algorithm are outlined in Algorithm 2. It is noted that there is a training stage before the application of Algorithm 2, where one performs dictionary learning and classifier training over suitable data training datasets. This algorithm should be superior to Algorithm 1. However, in situations where the priority is alleviating the computational burden of the classifier training and testing, this algorithm forms a compromise at the price of degraded performance.

The proposed algorithm assumes the availability of a dictionary $\boldsymbol{D}$ trained over sample received signals. Despite its high representation quality, learning a dictionary is a computationally demanding process, as will be discussed in the next session. This is especially the case knowing that SS will be typically performed by mobile users of limited computational and storage capabilities. Therefore, we argue that one can use a sample dictionary instead of a learned one. A sampled dictionary is obtained by randomly selected data vectors [27]. Compared to a learned dictionary, a sampled one is not as good in representation. However, it offers a cheap compromise for the computational complexity at a tolerable performance loss. For the problem of SS, using a sampled dictionary seems especially reasonable owing to the fact that the interest is not high-quality estimation as in other applications of sparse coding. Rather, the aim is to extract distinctive features from the received signals.

D. A DISCUSSION ON COMPUTATIONAL COMPLEXITY

In view of the stages of the proposed algorithm, it is clear that dictionary learning and classifier training incur the majority of the training stage computational cost. Correspondingly, the processes of sparse coding and classification occupy the major computational burden of the run-time stage. Accordingly, we can roughly approximate the proposed algorithm's computational complexity by addressing the complexities of these two processes.

First, let us address the computational complexity of classifier training, by considering SVM as an example. Given a training set consisting of a matrix $\boldsymbol{X} \in \mathbb{R}^{N\times L}$ representing the coordinates of m points in N dimensions and a target label vector $y \in \mathbb{R}^L$, O. Chapelle [28] reports that the computational complexity of SVM training depends on the relation between the dimensions L and N. This relation determines whether the primal or the dual SVM formulation is considered. Accordingly, the computational complexity of SVM training is $\mathcal{O}(\max\{L,N\}.\min\{L,N\}^2)$. On the other hand, dictionary learning is also required in the training stage of the proposed algorithm, if sampled dictionaries are not to be employed. Let us consider K-SVD [23] as an example. The total complexity of K-SVD working on a training set $\boldsymbol{X} \in C^{N\times L}$, with sparsity S and Num iterations is $\mathcal{O}(Num(S^2 + N)KL)$ [29]. It is noted that dictionary training is done off-line, only once.

To this end, we can address the computational complexities of SVM training and sparse coding. In [30], it is reported that the computational complexity of SVM testing (prediction) depends basically on the kernel used. For linear kernels, the computational complexity is in the order of the signal dimension. This means that for the computational complexity is $\mathcal{O}(n)$, where n is the dimension of the gradient vector to be fed to SVM prediction. It is noted that this work uses a linear kernel for SVM. This is due to the fact that the gradient vector values are clearly distinctive features for classification. Intuitively, there is no need to map these features to higher dimensional feature spaces by using a more advanced kernel, such as the Gaussian or polynomial kernels [31].

For sparse coding, let us consider OMP as an example. OMP has several computational approaches with different computational complexity and memory requirements. For the

TABLE 1. Computational complexity levels of the proposed algorithm with several operation scenarios.

	Computational Complexity of	
Operation Scenario	Training	Testing
Algorithm 1 with a learned $\boldsymbol{D}$	$\mathcal{O}(Num(S^2+N)KL)$	$\mathcal{O}(2KM^2+KM^3+M^4)$
Algorithm 1 with a sampled $\boldsymbol{D}$	N.A.	$\mathcal{O}(2KM^2+KM^3+M^4)$
Algorithm 2 with a learned $\boldsymbol{D}$	$\mathcal{O}(\max\{L,N\}.\min\{L,N\}^2)+Num(S^2+N)KL$	$\mathcal{O}(2KM^2+KM^3+M^4+M)$
Algorithm 2 with a sampled $\boldsymbol{D}$	$\mathcal{O}(\max\{L,N\}.\min\{L,N\}^2)$	$\mathcal{O}(2KM^2+KM^3+M^4+M)$

sake of convenience, we can consider the Naive OMP algorithm working on sparse coding of a signal $\boldsymbol{x} \in \mathbb{C}^N$ over a given dictionary $\boldsymbol{D} \in \mathbb{C}^{N\times K}$, with sparsity S. In [32], Sturm and Christensen prove that the k-th iteration of the naive OMP implementation has a computational complexity of $\mathcal{O}(NK + KS + KS^2 + S^3)$. The memory required for the naive approach is $\mathcal{O}(NK)$. This memory is required to store the dictionary and the inner products of an iteration. Hence, overall computational complexity with sparsity S will be $\mathcal{O}(NKS + KS^2 + KS^3 + S^4)$.

Based on the aforementioned discussion, the computational complexity of the proposed algorithm will be approximately as follows. For a given $\boldsymbol{y}_c \in \mathbb{C}^M$, $\boldsymbol{D} \in \mathbb{C}^{M\times K}$ with a compression factor of M, Algorithm 1 performs a sparse coding with a sparsity of M, where M is the number of measurements. So, the computational complexity of this algorithm is approximately $\mathcal{O}(2KM^2 + KM^3 + M^4)$.

The computational complexity of Algorithm 2 is similar to that of Algorithm 1 plus that of SVM prediction. Hence, it is approximately $\mathcal{O}(2KM^2 + KM^3 + M^4 + M)$. In summary, Table 1 lists the computational complexities of the proposed algorithm in these variants.

IV. SIMULATION AND RESULTS

In this section, we present performance analyses of the proposed algorithm as tested over synthetic and measured received signals. The performance metrics are the probability of detection P_D and false alarm rate P_F measures, defined as follows.

$$P_d = \Pr\{\text{decision} = \mathcal{H}_1 \mid \mathcal{H}_1\}, \tag{14}$$
$$P_F = \Pr\{\text{decision} = \mathcal{H}_1 \mid \mathcal{H}_0\}. \tag{15}$$

A. PARAMETERS SETTING

We use a system simulator created under MatLab environment. According to the system model specifications provided above, we simulate this system with different modulation settings. This includes phase shift keying (PSK), frequency shift keying (PSK), pulse amplitude modulation (PAM) or quadrature amplitude modulation (QAM) as modulation techniques. A complete listing of the simulation parameters is provided in Table 2. For each received signal realization, a different data stream, and a different channel realization are randomly obtained.

For the following experiments, a training set is made up of 10^4 received signal realizations generated synthetically to cover different SNR values. First, we create a training set of 10^4 example received signals. These are generated according to Table 2, and the modulation is randomly selected to be PSK, FSK, PAM, or QAM. Then an under-complete dictionary is trained for each modulation case. Next, the trained dictionaries are concatenated to establish a composite dictionary $\boldsymbol{D}$. The K-SVD algorithm [23] with standard settings is used for this purpose. We use a fixed sparsity of $S = 10$, 30 iterations and an overall dictionary size of 100×100. Then, another 10^4 received signals are generated to randomly be in $\mathcal{H}_0$ and $\mathcal{H}_1$. These will serve as the test set.

TABLE 2. Synthetic received signal simulation parameters.

Property	Value
Channel Model	Rayleigh
No. of Channel Taps	7
Channel Delay Unit	Sample Period
Signal Length	100
Oversampling Rate	10
Modulation Order	64
Symbol Order	Binary
Pulse Shaping	Square-root-raised-cos.
Raised Cos. Roll-off Factor	0.2
Raised Cos. Symbol Span	50

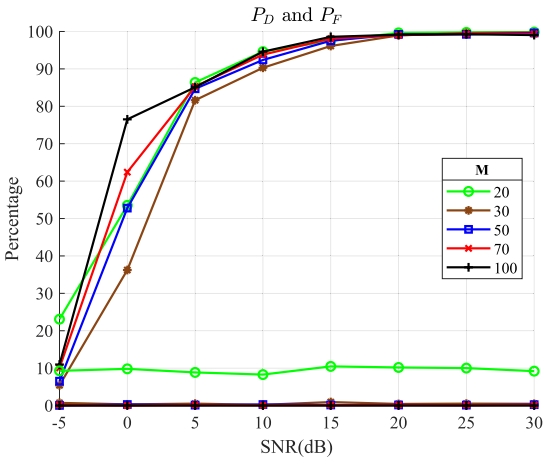


FIGURE 4. For Algorithm 1 with a learned dictionary, P_D and P_F versus SNR for several M values.

The following experiments present a performance analysis of the proposed algorithm. Both cases of synthetically generated and measured received signals are considered.

B. ROC PERFORMANCE WITH SYNTHETIC SIGNALS

1) WITH LEARNED DICTIONARIES

Herein, we use the dictionary and test set presented earlier. Each test signal is randomly selected to belong either to $\mathcal{H}_0$ or to $\mathcal{H}_1$. For each test signal, we obtain a compressed

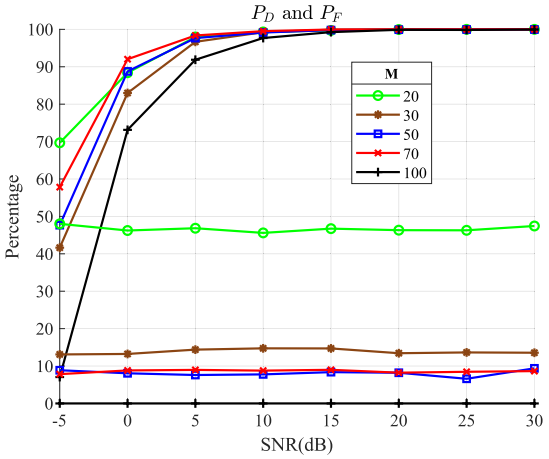


FIGURE 5. For Algorithm 1 with a sampled dictionary, P_D and P_F versus SNR for several M values.

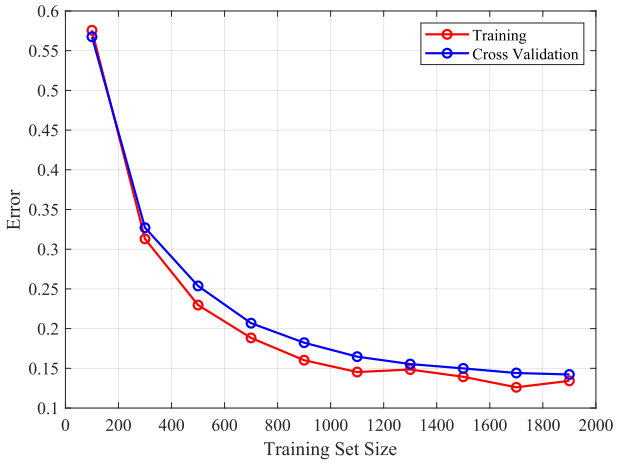


FIGURE 6. The SVM learning curve.

TABLE 3. Confusion matrices of SVM (above) and DNN (below), versus M.

	M									
	20		30		50		70		100	
SVM	95	5	98	2	100	0	100	0	100	0
	7	93	4	96	3	97	2	98	6	94
DNN	95	5	94	6	95	5	100	0	99	1
	13	87	9	91	12	88	2	98	13	87

version according to (1) using a Gaussian random sensing matrix. We detect the PU existence using Algorithm 1. Then, we calculate the average P_D and P_F measures over the given 10^4 test set signals. This procedure is repeated for several numbers of samples (M) over several SNR levels. The results are presented in Fig. 4. It is noted that $M = 100$ corresponds to the case where the sensing matrix is the identity matrix. Therefore, this value resembles sampling at the Nyquist rate.

In view of Fig. 4, the following observations can be made regarding the performance of the proposed algorithm. First, the P_D and P_F performances are both weak for the particular case of $SNR = 5dB$. However, the P_D performance is generally high for high M values. For $M > 30$, P_D is consistently greater than 50%. Second, the P_D performance also depends on the SNR value. However, for SNR values of $5dB$ and beyond, P_D is very high. Third, the P_F is moderate for the case of $M = 20$. For higher M values, P_F is consistently close to zero, regardless of the SNR.

To this end, it is interesting to investigate the performance of Algorithm 2 with SVM and DNN classifiers. We consider linear SVM and a 10-layer DNN. Each classifier model is trained over 1000 training vectors according to Fig 3. This number is selected as a compromise between computational complexity and performance trade-off in view of the learning curve [33] shown in Fig. 6. This curve shows the DNN cross-validation error versus training set size averaged over 100 trials. It is noted that SVM is set to use the same number for fair compassion. This curve shows an approximate good fit, as the performance of the model is good on both the train and validation sets.

A confusion matrix is used as an indication of the properties of a classification (discriminant) rule. It contains the number of elements that have been correctly or incorrectly classified for each class. We can see on its main diagonal the number of observations that have been correctly classified for each class; the off-diagonal elements indicate the number of observations that have been incorrectly classified. The rows of the confusion matrix correspond to the true class and the columns correspond to the predicted class. Diagonal and off-diagonal cells correspond to correctly and incorrectly classified observations, respectively. To attest the quality of the classifier learning, Table 3 lists the confusion matrices for different M values with the cases of SVM and DNN, respectively. It is seen that SVM is generally better than DNN in terms of the confusion matrix. Besides, the confusion matrices in both scenarios become better with increasing M.

The previous experiment is repeated while the decision is made this time according to Algorithm 2 working with either SVM or DNN. The results of using SVM and DNN are depicted in Figs. 7 and Fig. 8, respectively. Comparing Fig. 7 to Fig. 4, the advantage of SVM classification is notable and clear. In Fig. 7, the P_D performance is steadily better. This is especially the case for low SNR values. Besides, the P_F performance is also better, especially for SNR values of 5 dB and above. For SNR values of 0 and 5 dB, P_F is first relatively high, before it approaches very small values after 5 dB SNR. Similar to the results of Fig. 4, the P_D performance goes higher with increased M values. If one compares the performances of SVM and DNN, it is seen that DNN does not have any advantage over SVM. This is the case for both the P_D and P_F measures.

2) WITH SAMPLED DICTIONARIES

Now, it is interesting to investigate the impact of using sampled dictionaries instead of learned ones. The results for using sampled dictionaries with Algorithm 1, Algorithm 2 with SVM, and Algorithm 2 with DNN are depicted in Fig.s 5 and 9, and 10, respectively.

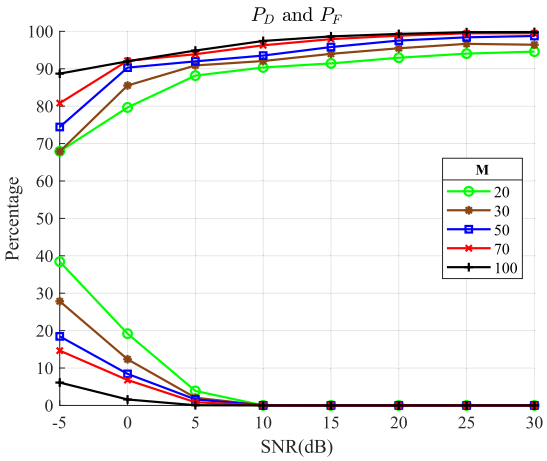


FIGURE 7. For algorithm 2 using SVM with a learned dictionary, P_D and P_F versus SNR for several M values.

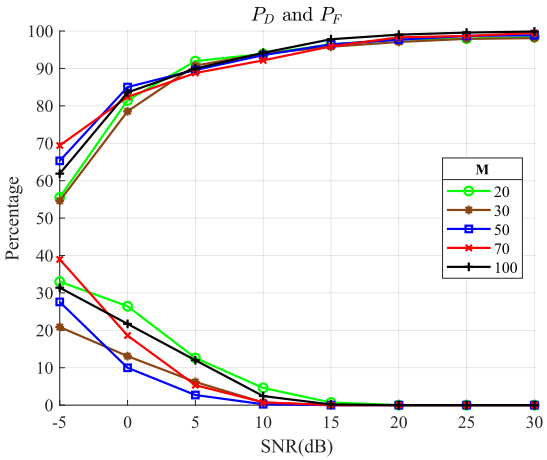


FIGURE 8. For algorithm 2 using DNN with a learned dictionary, P_D and P_F versus SNR for several M values.

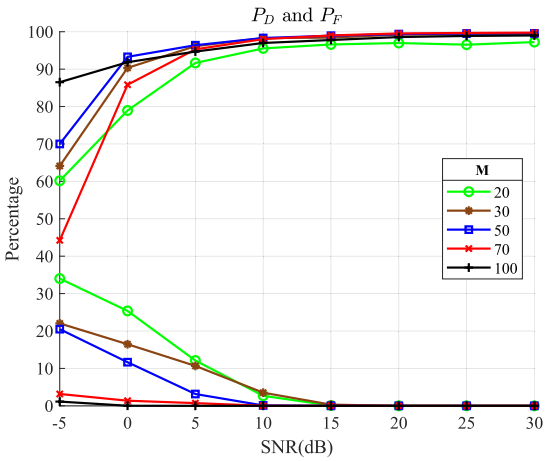


FIGURE 9. For algorithm 2 using SVM with a sampled dictionary, P_D and P_F versus SNR for several M values.

Comparing Fig. 5 to Fig. 4, the P_D performance is slightly better. On the contrary, the P_F performance is more significantly degraded. In summary, the P_F performance

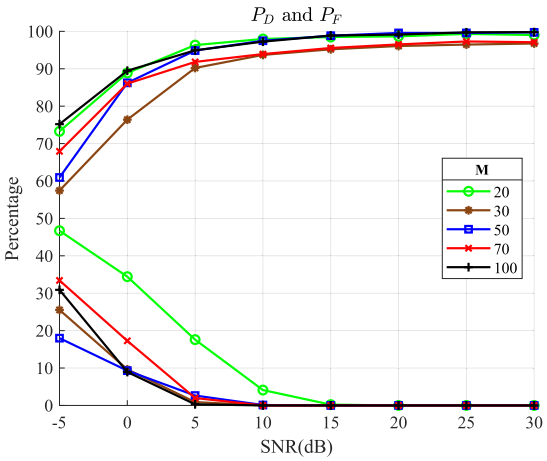


FIGURE 10. For algorithm 2 using DNN with a sampled dictionary, P_D and P_F versus SNR for several M values.

degradation is higher than the P_D performance gain. On the other hand, it is clear that a sampled dictionary slightly degrades both P_D and P_F performance of Algorithm 2, as seen by comparing Fig. 9 to Fig. 7. Moreover, Fig. 10 suggests that using DNN gives almost the same performance attained with SVM. However, it exhibits higher P_F values. This result is consistent with the comparison of SVM and DNN for the case of using learned dictionaries.

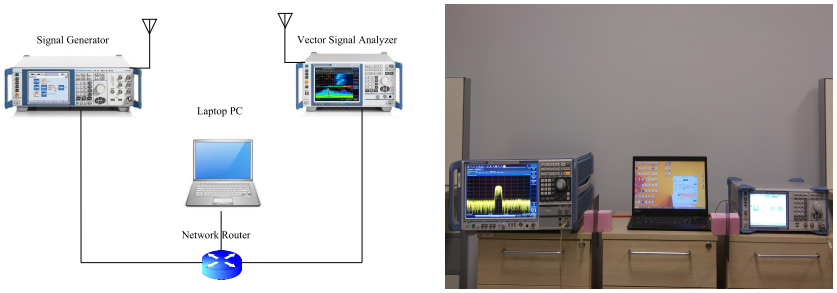
C. ROC PERFORMANCE WITH MEASURED SIGNALS

This test analyses the performance of the proposed algorithm as tested over indoor lab measurements of received signals. We use two sets of measurements; a training set and a test set. Such measurements were conducted with an SNR of -5 dB. To take these measurements, we adopted the same experimental setup presented in [34]. This experimental setup is composed of a Rohde & Schwarz SMBV100A vector signal generator, a Rohde & Schwarz FSW signal and spectrum analyzer, a tuple of omnidirectional antennas to cover the wireless communications bands of interest, a laptop computer, cables, and connectors, as depicted in Fig. 11.

TABLE 4. The values of the parameters used in practical signal measurements.

Property	Value Used
Carrier Frequency	918 MHz
Tx/ Rx Distance	18.24 Feet
Modulation Type	PSK, QAM, BPSK, QPSK
Modulation Order	8, 16, 64
Propagation Type	LOS/ NLOS
Filter Type	Square-root raised cosine

The signal generation, transmission, and measurement processes are controlled by the computer through a MatLab-based software. This software orders the signal generator to produce signals with various modulation schemes and parameters. These include PSK, FSK, QAM, and PAM modulated signals with various modulation orders based on the signal model in Section II. The list of all the parameters used in the measurement process is given in Table 4.



(a) (b)

FIGURE 11. Measurement setup and environment. (a) Measurement setup: transmitter, receiver, laptop computer, router, and their connections. (b) General overview of the measurement environment.

TABLE 5. ROC percentage with algorithm 1 over measured received signals.

	M					
Metric	10	20	30	50	70	100
P_D	100	100	100	100	100	100
P_F	0	0	0	0	0	0

TABLE 6. ROC percentage with algorithm 2 with SVM over measured received signals.

	M					
Metric	10	20	30	50	70	100
P_D	100	100	100	100	100	100
P_F	0	0	0	0	0	0.05

First, we consider Algorithm 2 with a learned dictionary and SVM classification. The P_D and P_F values are presented in Table 5. It is seen that Algorithm 2 provides excellent performance in both measures. Finally, we repeat this experiment with a sampled dictionary and provide the results in Table 6. Again, the proposed algorithm archives a full P_D with zero P_F consistently for all M values considered.

V. CONCLUSION

This paper has shown the possibility of exploiting the convergence features of sparse recovery as a means of spectrum sensing. Sparse recovery is applied to reconstruct a received signal from its compressed version obtained through a compressive sampling process. This sampling scheme allows for sub-Nyquist sampling, thereby reducing the analog-to-digital conversion burden. A learned dictionary is used for the recovery process. While doing sparse recovery, we quantify the reconstruction convergence speed in terms of the decay rate of the energy of sparse coding residual components. In this context, fast residual decay in the first few iterations signifies the existence of a transmitted signal. We also extend the work by using machine learning classification which uses the gradient of energy decay as a classification feature. This classification enhances the performance of the proposed algorithm. The proposed algorithm is shown to have excellent performances in terms of the probability-of-detection and false-alarm-rate measures. This result is validated through experiments conducted over synthetic data as well as real-life measurements of received signals.

The proposed algorithm can be extended to the wideband SS scenario. This can be achieved by applying the proposed algorithm on several subbands of the frequency range of interest. This requires designing several dictionaries that can account for the whole spectrum of interest.

APPENDIX
RESIDUAL ENERGY GRADIENT DECAY ANALYSIS

By tracing sparse coding atom selection and residual update, one can roughly analyze how the decay rate of the residual energy is related to the existence of a transmitted signal. Let us assume that OMP is the sparse coding algorithm. Now, let us concentrate on the first OMP iteration for the sake of convenience.

At the beginning of the first iteration, OMP initializes the zero-th residual $\boldsymbol{r}_0$ by the signal of interest. Then, amongst all atoms in a given dictionary $\boldsymbol{D}$, the one that best approximates $\boldsymbol{r}_0$ based on a certain similarity measure, is selected. OMP considers maximizing the projection of $\boldsymbol{r}_0$ over each atom. This is achieved by calculating a projection operator for each atom $\boldsymbol{d}$ as $\boldsymbol{P} = dd^{\dagger}$, where $\dagger$ is the Moore-Penrose pseudoinverse. After this atom selection, the residual is updated as follows

$$\boldsymbol{r}_1 = \boldsymbol{r}_0 - \boldsymbol{P}\boldsymbol{r}_0. \tag{16}$$

Let us ignore the least-squares refinement OMP does for the sake of simplicity.

Now, we can write the gradient operator $\boldsymbol{G}$ as the discrete first derivative of the residual energy. Hence, the first element in $\boldsymbol{G}$ can be expressed as follows.

$$\boldsymbol{G}(1) = \|\boldsymbol{r}_1\|_2^2 - \|\boldsymbol{r}_0\|_2^2 = \langle \boldsymbol{r}_1, \boldsymbol{r}_1 \rangle - \langle \boldsymbol{r}_0, \boldsymbol{r}_0 \rangle. \tag{17}$$

Let us analyze the gradient magnitude comparing the cases of $\mathcal{H}_0$ and $\mathcal{H}_1$.

Under $\mathcal{H}_0$: the signal $\boldsymbol{y}$ is merely noise. Therefore, $\boldsymbol{G}(1)$ can be written as follows.

$$\begin{aligned} \boldsymbol{G}(1) &= \|\boldsymbol{n} - \boldsymbol{P}\boldsymbol{n}\|_2 - \|\boldsymbol{n}\|_2 \\ &= \langle \boldsymbol{n} - \boldsymbol{P}\boldsymbol{n}, \boldsymbol{n} - \boldsymbol{P}\boldsymbol{n} \rangle - \langle \boldsymbol{n}, \boldsymbol{n} \rangle. \end{aligned} \tag{18}$$

Note that $\langle \boldsymbol{n} - \boldsymbol{P}\boldsymbol{n}, \boldsymbol{n} - \boldsymbol{P}\boldsymbol{n} \rangle - \langle \boldsymbol{n}, \boldsymbol{n} \rangle = \langle \boldsymbol{n}, \boldsymbol{n} \rangle - 2\langle \boldsymbol{n}, \boldsymbol{P}\boldsymbol{n} \rangle + \langle \boldsymbol{P}\boldsymbol{n}, \boldsymbol{P}\boldsymbol{n} \rangle - \langle \boldsymbol{n}, \boldsymbol{n} \rangle$. Now, using the property of projection

we know that $\langle \boldsymbol{Pn}, \boldsymbol{n} \rangle = \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle = \|\boldsymbol{Pn}\|_2^2$, we can write

$$\boldsymbol{G}(1)_{\mathcal{H}_0} = -\|\boldsymbol{Pn}\|_2^2. \tag{19}$$

Under $\mathcal{H}_1$: the signal $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$, where the channel operator ($\boldsymbol{H}$) is dropped from simplicity. Therefore, $\boldsymbol{G}(1)$ can be written as follows. $\boldsymbol{G}(1) = \langle \boldsymbol{x}+\boldsymbol{n} - \boldsymbol{P}(\boldsymbol{x}+\boldsymbol{n}), \boldsymbol{x}+\boldsymbol{n} - \boldsymbol{P}(\boldsymbol{x}+\boldsymbol{n}) \rangle - \langle \boldsymbol{x}+\boldsymbol{n}, \boldsymbol{x}+\boldsymbol{n} \rangle = a - 2b + c - d$. Let us simplify the four terms a, b, c, and d.

First: $a = \langle \boldsymbol{x}+\boldsymbol{n}, \boldsymbol{x}+\boldsymbol{n} \rangle = \langle \boldsymbol{x}+\boldsymbol{x} \rangle + 2\langle \boldsymbol{x}+\boldsymbol{n} \rangle + \langle \boldsymbol{n}+\boldsymbol{n} \rangle$ assuming that the noise is independent of $\boldsymbol{x}$, $\langle \boldsymbol{x}+\boldsymbol{n} \rangle = 0$ we can write $a = \langle \boldsymbol{x}+\boldsymbol{x} \rangle + \langle \boldsymbol{n}+\boldsymbol{n} \rangle$.

Second: $b = \langle \boldsymbol{x}+\boldsymbol{n}, \boldsymbol{P}(\boldsymbol{x}+\boldsymbol{n}) \rangle = \langle \boldsymbol{x}+\boldsymbol{n}, \boldsymbol{Px}+\boldsymbol{Pn} \rangle = \langle \boldsymbol{x}, \boldsymbol{Px} \rangle + \langle \boldsymbol{x}, \boldsymbol{Pn} \rangle + \langle \boldsymbol{n}, \boldsymbol{Px} \rangle + \langle \boldsymbol{n}, \boldsymbol{Pn} \rangle$. Using the expression: $\langle \boldsymbol{x}, \boldsymbol{Px} \rangle = \langle \boldsymbol{Px}, \boldsymbol{Px} \rangle$, $\langle \boldsymbol{n}, \boldsymbol{Pn} \rangle = \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle$, and $\langle \boldsymbol{x}, \boldsymbol{Pn} \rangle = \langle \boldsymbol{n}, \boldsymbol{Px} \rangle = 0$, we can write: $b = \langle \boldsymbol{Px}, \boldsymbol{Px} \rangle + \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle$

Third: $c = \langle \boldsymbol{P}(\boldsymbol{x}+\boldsymbol{n}), \boldsymbol{P}(\boldsymbol{x}+\boldsymbol{n}) \rangle = \langle \boldsymbol{Px}+\boldsymbol{Pn}, \boldsymbol{Px}+\boldsymbol{Pn} \rangle = \langle \boldsymbol{Px}, \boldsymbol{Px} \rangle + \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle$

Fourth: $d = a = \langle \boldsymbol{x}+\boldsymbol{x} \rangle + \langle \boldsymbol{n}+\boldsymbol{n} \rangle = a$. Therefore, $\boldsymbol{G}(1) = a - 2b + c + d = -2\langle \boldsymbol{Px}, \boldsymbol{Px} \rangle - 2\langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle + \langle \boldsymbol{Px}, \boldsymbol{Px} \rangle + \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle = -\langle \boldsymbol{Px}, \boldsymbol{Px} \rangle - \langle \boldsymbol{Pn}, \boldsymbol{Pn} \rangle$

Finally, the gradient magnitude for $\mathcal{H}_1$ is as shown in (20).

$$\boldsymbol{G}(1)_{\mathcal{H}_1} = -\|\boldsymbol{Px}\|_2 - \|\boldsymbol{Pn}\|_2. \tag{20}$$

Comparing (19) and (20), it is evident that $\boldsymbol{G}(1)_{\mathcal{H}_1}$ is greater than $\boldsymbol{G}(1)_{\mathcal{H}_0}$ as anticipated earlier. Following similar steps, this result can be generalized for the first S iterations of OMP.

REFERENCES

[1] M. A. McHenry, "NSF spectrum occupancy measurements project summary," *Shared Spectr. Company Rep.*, Aug. 2005.

[2] J. Yang, M. Jia, X. Gu, and Q. Guo, "Low complexity sub-nyquist wideband spectrum sensing for cognitive radio," *IEEE Access*, vol. 6, pp. 45166–45176, 2018.

[3] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 201–220, Feb. 2005.

[4] H. V. Poor, *An Introduction to Signal Detection and Estimation*. Springer, 2013.

[5] H. Sun, D. I. Laurenson, and C.-X. Wang, "Computationally tractable model of energy detection performance over slow fading channels," *IEEE Commun. Lett.*, vol. 14, no. 10, pp. 924–926, Oct. 2010.

[6] J. Chen, A. Gibson, and J. Zafar, "Cyclostationary spectrum detection in cognitive radios," in *Proc. IET Seminar Cogn. Radio Softw. Defined Radios, Technol. Techn.*, Sep. 2008, pp. 1–5.

[7] X. Zhang, Y. Ma, Y. Gao, and S. Cui, "Real-time adaptively regularized compressive sensing in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1146–1157, Feb. 2018.

[8] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1527–1550, 3rd Quart., 2017.

[9] Z. Gao, L. Dai, S. Han, C.-L. I, Z. Wang, and L. Hanzo, "Compressive sensing techniques for next-generation wireless communications," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 144–153, Jun. 2018.

[10] H. Qi, X. Zhang, and Y. Gao, "Channel energy statistics learning in compressive spectrum sensing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 7910–7921, Dec. 2018.

[11] M. A. Davenport, P. T. Boufounos, M. B. Wakin, and R. G. Baraniuk, "Signal processing with compressive measurements," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 445–460, Apr. 2010.

[12] H. Sun, A. Nallanathan, C.-X. Wang, and Y. Chen, "Wideband spectrum sensing for cognitive radio networks: A survey," *IEEE Wireless Commun.*, vol. 20, no. 2, pp. 74–81, Apr. 2013.

[13] J. Jiang, H. Sun, D. Baglee, and H. V. Poor, "Achieving autonomous compressive spectrum sensing for cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1281–1291, Mar. 2016.

[14] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Syst. Comput.*, Nov. 1993, pp. 40–44.

[15] Z. Qin, J. Fan, Y. Liu, Y. Gao, and G. Y. Li, "Sparse representation for wireless communications: A compressive sensing approach," *IEEE Signal Process. Mag.*, vol. 35, no. 3, pp. 40–58, May 2018.

[16] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 33, no. 1, pp. 33–61, 2001.

[17] C. Zhang, D. Hao, C. Hou, and X. Yin, "A new approach for sparse signal recovery in compressed sensing based on minimizing composite trigonometric function," *IEEE Access*, vol. 6, pp. 44894–44904, 2018.

[18] Z. Qin, Y. Gao, M. D. Plumbley, and C. G. Parini, "Wideband spectrum sensing on real-time signals at sub-Nyquist sampling rates in single and cooperative multiple nodes," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3106–3117, Jun. 2016.

[19] Y. Wang, Z. Tian, and C. Feng, "Sparsity order estimation and its application in compressive spectrum sensing for cognitive radios," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 2116–2125, Jun. 2012.

[20] H. Sun, W.-Y. Chiu, and A. Nallanathan, "Adaptive compressive spectrum sensing for wideband cognitive radios," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1812–1815, Nov. 2012.

[21] Z. Qin, Y. Gao, and C. G. Parini, "Data-assisted low complexity compressive spectrum sensing on real-time signals under sub-Nyquist rate," *IEEE Trans. Wireless Commun.*, vol. 15, no. 2, pp. 1174–1185, Feb. 2016.

[22] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[23] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[24] T. Zhang, "An introduction to support vector machines and other kernel-based learning methods," *AI Mag.*, vol. 22, no. 2, p. 103, 2001.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[27] Y. Jianchao, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[28] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, May 2007.

[29] K. Skretting and K. Egan, "Recursive least squares dictionary learning algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2121–2130, Apr. 2010.

[30] G. Sharma, F. Jurie, and P. Perez, "Learning non-linear SVM in input space for image classification," Ph.D. dissertation, CNRS-GREYC UMR, Univ. de Caen, Caen, France, 2014.

[31] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. (2013). *A practical Guide to Support*. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[32] B. L. Sturm and M. G. Christensen, "Comparison of orthogonal matching pursuit implementations," in *Proc. 20th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2012, pp. 220–224.

[33] C. Perlich, "Learning curves in machine learning," IBM, Armonk, NY, USA, Res. Rep. RC24756 (W0903-020), Mar. 2009.

[34] A. Gorcin and H. Arslan, "An OFDM signal identification method for wireless communications systems," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5688–5700, Dec. 2015.

MAHMOUD NAZZAL received the B.Sc. degree in electrical engineering from Birzeit University, in 2009, and the M.Sc. and Ph.D. degrees in electrical and electronic engineering from Eastern Mediterranean University, in 2010 and 2015, respectively, where he was a Lecturer with the Electrical and Electronic Engineering Department, from 2015 to 2016. He was also a Lecturer with the Izmir University of Economics, from 2016 to 2017. Since July 2017, he has been a Postdoctoral Researcher with Istanbul Medipol University. His research interests include sparse coding, compressive sensing, computer vision, and signal processing for wireless communications.

ALİ RIZA EKTİ was born in Tarsus, Turkey. He received the degree from Universidad Politechnica de Valencia, Valencia, Spain, in 2005, the B.Sc. degree in electrical and electronics engineering from Mersin University, Mersin, Turkey, in June 2006, the M.Sc. degree in electrical engineering from the University of South Florida, Tampa, Florida, in December 2009, and the Ph.D. degree in electrical engineering from the Department of Electrical Engineering and Computer Science, Texas A&M University, in August 2015. He is currently an Assistant Professor with the Electrical and Electronics Engineering Department, Balıkesir University, and also a Senior Researcher with BILGEM, TÜBITAK. His current research interests include statistical signal processing, convex optimization, machine learning, resource allocation and traffic offloading in wireless communications in 4G and 5G systems, and smart grid design and optimization.

ALİ GÖRÇİN received the B.Sc. degree in electronics and telecommunications engineering and the master's degree in defense technologies from Istanbul Technical University, and the Ph.D. degree in wireless communications with the University of South Florida (USF), USA. He was with the Turkish Science Foundation (TÜBITAK) on avionics projects for more than six years. He was with Anritsu Company during his tenure in USF, also with Reverb Networks and Viavi Solutions after his graduation. He is currently an Assistant Professor with Yıldız Technical University, Istanbul. He is also serving as the Vice President of the Informatics and Information Security Research Center, TÜBITAK, responsible for testing and evaluation along with research and development activities on wireless communications technologies.

HÜSEYİN ARSLAN (S'95–M'98–SM'04–F'15) received the B.S. degree from Middle East Technical University, Ankara, Turkey, in 1992, and the M.S. and Ph.D. degrees from Southern Methodist University, Dallas, TX, USA, in 1994 and 1998, respectively. From 1998 to 2002, he was with the Research Group, Ericsson, Inc., NC, USA, where he was involved with several projects related to 2G and 3G wireless communication systems. He was a part-time consultant for various companies and institutions, including Anritsu Company, Morgan Hill, CA, USA, and the Scientific and Technological Research Council of Turkey. Since 2002, he has been with the Electrical Engineering Department, University of South Florida, Tampa, FL, USA. He has also been the Dean of the College of Engineering and Natural Sciences, Istanbul Medipol University, since 2014. His research interests include physical-layer security, millimeter-wave communications, small cells, multicarrier wireless technologies, co-existence issues on heterogeneous networks, aeronautical (high-altitude platform) communications, *in vivo* channel modeling, and system design.

He is currently a member of the Editorial Board of the IEEE SURVEYS AND TUTORIALS AND the SENSORS Journal. He also served as a member of the Editorial Board of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, the Elsevier *Physical Communication* Journal, the Hindawi *Journal of Electrical and Computer Engineering*, and the Wiley *Wireless Communication and Mobile Computing* Journal. He served as the Technical Program Committee Chair, a Technical Program Committee Member, a Session and Symposium Organizer, and the Workshop Chair for several IEEE conferences.