IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 November 2022, accepted 19 November 2022, date of publication 24 November 2022,
date of current version 6 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3224478

RESEARCH ARTICLE

Enumeration and Identification of Active Users for Grant-Free NOMA Using Deep Neural Networks

MUHAMMAD USMAN KHAN, (Graduate Student Member, IEEE), ENRICO PAOLINI, (Senior Member, IEEE), AND MARCO CHIANI, (Fellow, IEEE)
CNIT/WiLab, DEI, University of Bologna, 40126 Bologna, Italy
Corresponding author: Marco Chiani (marco.chiani@unibo.it)

ABSTRACT In next-generation mobile radio systems, multiple access schemes will support a massive number of uncoordinated devices exhibiting sporadic traffic, transmitting short packets to a base station. Grant-free non-orthogonal multiple access (NOMA) has been introduced to provide services to a large number of devices and to reduce the communication overhead in massive machine-type communication (mMTC) scenarios. In grant-free communication, there is no coordination between the device and base station (BS) before the data transmission; therefore, the challenging task of active users detection (AUD) must be conducted at the BS. For NOMA with sparse spreading, we propose a deep neural network (DNN)-based approach for AUD called active users enumeration and identification (AUEI). It consists of two phases: firstly, a DNN is used to estimate the number of active users; then in the second phase, another DNN identifies them. To speed up the training process of the DNNs, we propose a multi-stage transfer learning technique. Our numerical results show a remarkable performance improvement of AUEI in comparison to previously proposed approaches.

INDEX TERMS Active user detection, deep neural network, grant-free, massive machine-type communication, non-orthogonal multiple access, transfer learning.

I. INTRODUCTION

In recent years, mMTC has gained a lot of attention due to applications such as smart grid and metering, smart factories, autonomous driving, and public health [1], [2]. In cellular scenarios, mMTC has to provide connectivity between BSs and a very large number of devices [3].

In a conventional multiple-access scenario consisting of a relatively small number of human-type users, the BS assigns radio resources in a coordinated fashion to each user. On the contrary, in mMTC scenario, the resource allocation approach will yield tremendous control signaling overhead which may be large in comparison to the size of the data, making the protocol highly inefficient.

To cope with these limitations, grant-free-based approaches have been proposed. In grant-free random access, signalling overhead and latency are reduced as the active devices transmit data without a grant procedure. In contrast to orthogonal multiple access, NOMA permits sharing of the same time-frequency resources, therefore, it can support a massive number of devices in a limited radio spectrum. In the code domain NOMA, each user is assigned a sparse spreading sequence, known to the BS. The length of the spreading sequences is kept low to efficiently utilize the radio spectrum. Due to a large number of users, the sequences are non-orthogonal. Despite this, decoding is possible in mMTC because the number of active devices at any given time is a small fraction of the total number of devices. Since there is no previous coordination or grant procedure, the BS must identify the active users to be able to decode them by their respective spreading sequences. Thus, the first crucial step is active user detection. Due to the sparseness of the users' activation pattern, compressive sensing (CS)-based techniques have been proposed in NOMA to identify

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

them [4], [5], [6]. In [7], the authors proposed a low-complexity algorithm for active users detection using pilot sequences with a massive number of antennas at the BS. A receiver which works independently of parameters such as signal-to-noise ratio (SNR) and user activity ratio in a NOMA setting is proposed in [8]. However, it has been shown that the performance of CS-based detection schemes degrade considerably as the sparsity level (number of active devices) increases [6]. Moreover, CS-based algorithms fail to consider time constraint [9]. For instance, the number of iterations of block iterative hard thresholding (BIHT) presented in [10] depends on the sparsity level, i.e., the algorithm will take more time to converge as the sparsity level increases.

To overcome some of these issues, deep learning (DL) methods could be used instead of CS. Indeed, it has been shown that a DNN can learn a large number of piecewise smooth functions [11], and since then DL methods have been successfully proposed in various fields, such as speech recognition [12], computer vision [13], and language translation [14]. DL techniques find several applications in the wireless communication domain as well [9], [15], [16], [17]. In contrast to CS solutions, DL requires a large amount of data for training, but once the algorithm is trained the complexity becomes low. Indeed, in the operational mode, DL involves multiply-accumulate and element-wise nonlinear evaluations, which are far less computationally expensive than the CS-based techniques [9], [18]. Thus, some studies have been carried out to identify active users in NOMA scenarios using DL algorithms [3], [18]. Specifically, a recurrent neural network (RNN) has been proposed for both AUD and channel estimation considering a NOMA scenario with sparse spreading sequences in [3]. Another approach that deals with AUD using a DNN architecture with residual connections has been proposed in [18]. The existing DNN-based algorithms for AUD can be divided into three categories: i) assuming the number of active users is perfectly known [19]; ii) without preliminary estimation of the number of active users [3], [20], [21]; iii) estimating this number through thresholding-based algorithms [18]. Assuming perfect knowledge of the number of active users is unrealistic. Also, sparsity estimation by thresholding-based algorithms is not an easy task, as the threshold level would depend on several system parameters in an unknown way, leading to poor results when compared with the other categories [3].

In this paper, we assume that at the beginning of the transmission the BS is unaware of the number of active users. The main contributions of this paper are summarized as follows

- we propose a new solution to active users detection comprised of two novel DNN architectures, one for sparsity estimation called active users enumeration (AUE), and the other one for identifying the active users called active users identification (AUI);
- we compare our solution with previous approaches to assess the performance improvement;
- we also report the false alarm rate to completely characterize the performance of our model. The false alarm rate has never been analyzed in the literature on AUD to the best of our knowledge;
- we investigate a multi-stage transfer learning approach to reduce the training time of the DNNs.

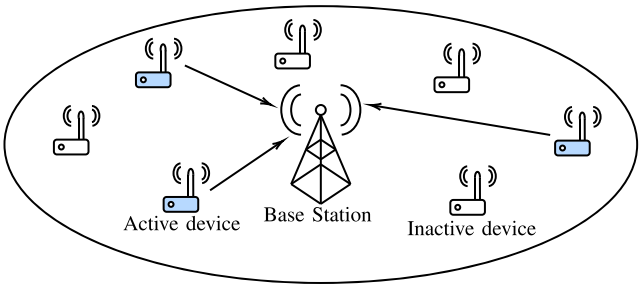


FIGURE 1. Depiction of GF-NOMA uplink communication scenario where only a few devices are active.

The rest of the paper is organized as follows. We present the system model along with the concept of spreading sequences and multiple measurements in Section II. In Section III, we explain the DNN architecture for AUD and sparsity estimation. Section IV contains the simulation settings and results. Section V concludes the study.

We use boldface uppercase, boldface lowercase, and lowercase letters to denote matrices, vectors, and scalars respectively. Also, abs(v) and arg(v) denote the magnitude and argument of the complex number v, respectively. The operator diag($\boldsymbol{v}$) outputs a diagonal matrix with entries of the vector $\boldsymbol{v}$ along the diagonal, and $\| \cdot \|_p$ represents the p-norm.

II. SYSTEM MODEL

We consider a synchronized uplink grant-free NOMA system scenario as in [18] and [3], in which N machine-type devices can transmit to the BS (see Fig. 1), both machine-type devices and BS are equipped with a single antenna, and each device is assigned a preconfigured sequence (or codeword), known by the BS. A small number of devices K are active at a given time, with $1 \leq K \leq K_{\max}$ and $K_{\max} \ll N$, where $K_{\max}$ is a system parameter representing the maximum number of active users under consideration. The symbols generated by each active device are spread with its device-specific non-orthogonal codeword. Then, the samples are transmitted through parallel frequency-flat channels, e.g., by Orthogonal Frequency-Division Multiplexing (OFDM).

For instance, if the i^{th} device wants to send at time t a symbol $s_i^{(t)} \in \mathbb{C}$, it encodes it into $\boldsymbol{q}_i^{(t)} = \boldsymbol{c}_i^{(t)} s_i^{(t)} \in \mathbb{C}^S$, where $\boldsymbol{c}_i^{(t)} = [c_{i,1}^{(t)}, \cdots, c_{i,S}^{(t)}]^T \in \mathbb{C}^S$ is the codeword of length S associated with the i^{th} device. The S elements of $\boldsymbol{q}_i^{(t)}$ are sent over S parallel additive white Gaussian noise (AWGN) channels with gains $\boldsymbol{h}_i^{(t)} = [h_{i,1}^{(t)}, h_{i,2}^{(t)}, \ldots, h_{i,S}^{(t)}]^T$. Overall, the received vector at the BS at time t can be written as

$$\boldsymbol{y}^{(t)} = \sum_{i=1}^{N} \delta_i \, \mathrm{diag}(\boldsymbol{c}_i^{(t)}) \boldsymbol{h}_i^{(t)} s_i^{(t)} + \boldsymbol{n}^{(t)} \tag{1}$$

where $\boldsymbol{n}$ denotes the complex Gaussian noise vector $\boldsymbol{n} \sim \mathcal{CN}(0, \sigma_n^2 \boldsymbol{I})$. The device indicator $\delta_i \in \{0, 1\}$ indicates the activity status of the i^{th} device, with $\delta_i = 0/1$ for inactive/active devices, respectively.

To minimize the interuser interference, we employ low-density signature (LDS) codewords, i.e., each codeword has only a small number n_S of non-zero values [22]. Similarly to [22], [18], and [3], to generate a codeword we first randomly pick n_S positions, and then generate the non-zero entries as independent and identically distributed (i.i.d.) according to a complex Gaussian distribution $\mathcal{CN}(0, \sigma_w^2)$.

Assuming the devices transmit N_d consecutive symbols, the received measurements can be arranged in a vector as follows

$$\tilde{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{\Phi}_1 \cdots \boldsymbol{\Phi}_N \end{bmatrix} \begin{bmatrix} \delta_1 \boldsymbol{x}_1 \\ \vdots \\ \delta_N \boldsymbol{x}_N \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}^{(1)} \\ \vdots \\ \boldsymbol{n}^{(N_d)} \end{bmatrix} \tag{2}$$

where $\boldsymbol{\Phi}_i = \text{diag}[(\boldsymbol{c}_i^{(1)})^T \cdots (\boldsymbol{c}_i^{(N_d)})^T]$ are the codebook matrices of dimension $(N_d \cdot S) \times (N_d \cdot S)$, $\boldsymbol{c}_i^{(t)}$ are the randomly generated codewords, and $\boldsymbol{x}_i = [(s_i^{(1)} \boldsymbol{h}_i^{(1)})^T \cdots (s_i^{(N_d)} \boldsymbol{h}_i^{(N_d)})^T]^T \in \mathbb{C}^{N_d \cdot S}$ denote the composite channel vectors and data symbols. For example, consider the case where only the 2^{nd} and 4^{th} users are active. Then, (2) reduces to

$$\tilde{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{\Phi}_2 \boldsymbol{\Phi}_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{x}_4 \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}^{(1)} \\ \vdots \\ \boldsymbol{n}^{(N_d)} \end{bmatrix}. \tag{3}$$

Assuming a maximum number of active users $K_{\max}$, AUD can be formulated as the support identification problem

$$\hat{\Omega} = \underset{\Omega,\, |\Omega| \le K_{\max}}{\arg\min} \; \| \tilde{\boldsymbol{y}} - \boldsymbol{\Phi}_\Omega \boldsymbol{x}_\Omega \|_2 \tag{4}$$

where Ω are the subsets of $\{1, 2, \ldots, N\}$, and the $\hat{\Omega}$ contains the indexes of the estimated active users.

One possible approach to solve (4) consists of applying CS-based techniques, which however could be challenging for real-time applications [6], [10], [23], [24], [25]. On the contrary, once a DNN is trained, estimating $\hat{\Omega}$ will be less computationally expensive with respect to CS-based approaches. In the next section, we will discuss our approach which is based on DL.

III. DEEP LEARNING-BASED AUD

Different approaches based on DNNs have been proposed in the literature for AUD, all employing thresholding-based algorithms for determining the number of active users [18], [26]. Here we present a different solution composed of two separate DNN architectures, one for active users enumeration and the other for active users identification. To the best of our knowledge, this is the first work which utilizes a DNN-based architecture for enumerating the active users in a NOMA scenario. The task of the AUE network is to output the number of active users, while a set of AUI networks, each trained for a different sparsity level, identifies the active users. More precisely, the former learns the mapping between the received vector $\tilde{\boldsymbol{y}}$ and the estimated number of active users $\hat{K}$, while the latter learns the mapping between the received vector $\tilde{\boldsymbol{y}}$ and $\hat{\Omega}$ for the cardinality $|\hat{\Omega}| = \hat{K}$. The networks provide the result as follows

$$\begin{aligned} \hat{K} &= f(\tilde{\boldsymbol{y}}; \Psi) \\ \hat{\Omega} &= g_{\hat{K}}(\tilde{\boldsymbol{y}}; \Theta_{\hat{K}}) \end{aligned} \tag{5}$$

where Ψ and Θ_k are the sets of weights and biases associated with the enumeration DNN and the identification DNN for sparsity $k \in \{1, 2, \ldots, K_{\max}\}$, respectively.

A. DNNs ARCHITECTURE

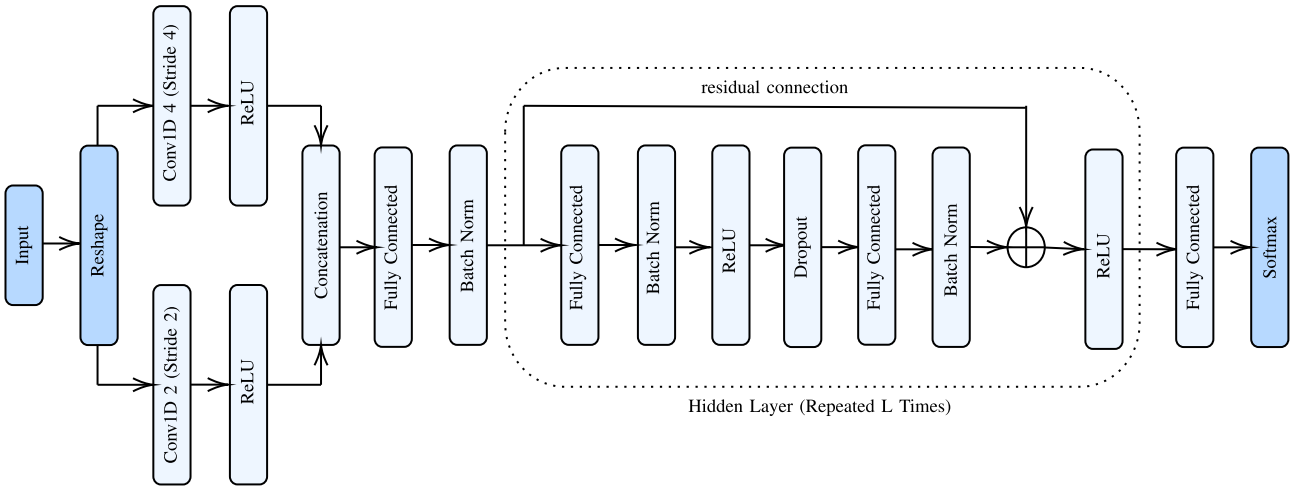
The received vector obtained through (2) has complex elements. To work with common DNNs, which assume real numbers as input, we split the magnitude and phase parts. More precisely, for a received vector $\tilde{\boldsymbol{y}} = [y_1, \ldots, y_m]^T \in \mathbb{C}^m$ then the input to the DNNs would be $\hat{\boldsymbol{y}} = [\text{abs}(y_1), \arg(y_1), \ldots, \text{abs}(y_m), \arg(y_m)]^T$.

Fig. 2 shows the architecture for the AUE and AUI. Both DNNs consist of convolutional layers, fully-connected layers, batch normalization layers, dropout layer, and activation layers. The difference between the AUE and AUI is in the output layer, which is a softmax for the AUE, and a sigmoid layer for the AUI. These output layers are described precisely below. The input to the DNNs $\hat{\boldsymbol{y}}$ is reshaped to a 2-D feature map $(N_d, 2S)$ using the reshape layer, where the first dimension corresponds to the channels analogous to the channels in a colour image. The 1-D convolution operation is performed using filters of size 2 and 4, with a stride equal to the filter size. We perform valid convolution, i.e., the output is only considered when the filter is fully contained in the feature map and the output feature map is reduced according to the input feature map, filter size and stride [27]. The output feature maps from the convolutional layers are passed through a Rectified Linear Unit (ReLU) activation function (described below). The output from the activation function is reduced to 1-D and then concatenated through the concatenation layer. The rationale behind using convolutional layers is to reduce the computational complexity and to extract the features shared among N_d multiple measurements. The fully connected layer, with input $\boldsymbol{a} \in \mathbb{R}^{in}$ and output $\boldsymbol{z} \in \mathbb{R}^{out}$, can be expressed as

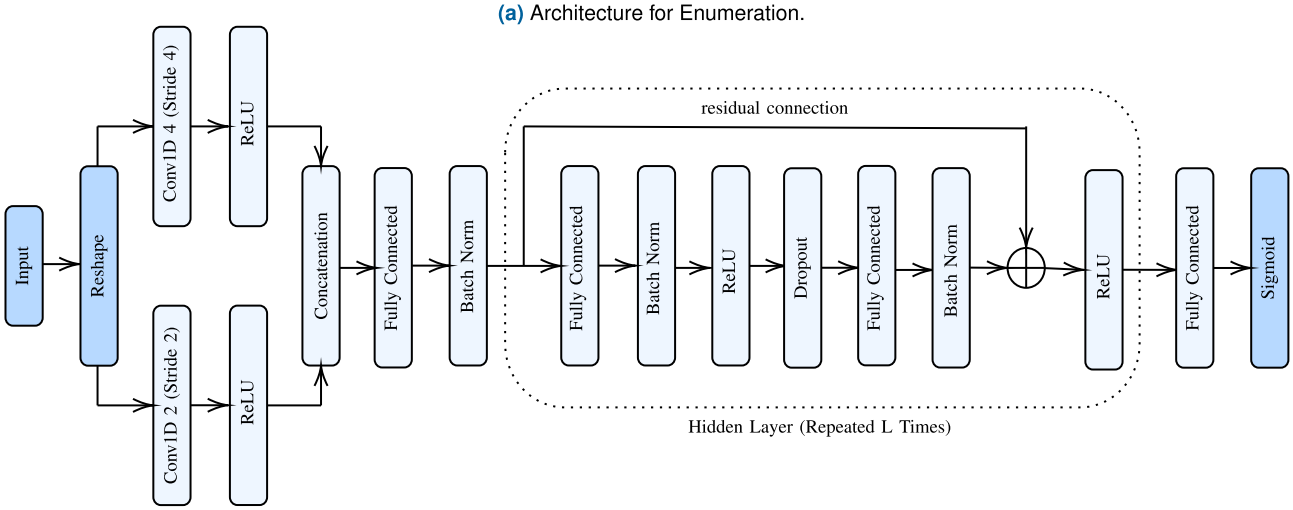
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b} \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^{out \times in}$ represents the weight matrix matching the dimension of output (out) and input (in) vector, and $\boldsymbol{b} \in \mathbb{R}^{out}$ describes the bias [27]. The fully-connected layers consist of α neurons, except for the last one. In fact, the last fully-connected layer dimension must agree with the output layer dimension, so it contains $K_{\max}$ and N neurons for the enumeration and the identification DNNs, respectively.

Instead of a single training example, we trained DNNs on a batch of training examples called a mini-batch, $\boldsymbol{B} = [\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(n)}]$. The batch normalization layer normalizes the



(a) Architecture for Enumeration.



(b) Architecture for Identification.

FIGURE 2. Architecture of the proposed DNNs.

mini-batch $\boldsymbol{B}$ to zero mean and unit variance, and then scales it using the trainable parameters γ and β

$$z_i^{(k)} = \frac{\gamma(a_i^{(k)} - \mu_i)}{\sqrt{\sigma_i^2}} + \beta \quad i = 1, \ldots, \alpha \tag{7}$$

where μ_i and σ_i^2 are estimates of the mean and variance of the i^{th} element of the vector, respectively, obtained by moving average [28]. The activation layers are introduced so that the DNNs can learn non-linear functions. ReLU is a common choice as an activation function in the hidden layers for numerous DNN architectures [27], [29] and ReLU can be mathematically described as

$$a = \max(0, z) \tag{8}$$

where the operation is to be considered element-wise.

A DNN consists of many hidden layers, and it becomes challenging to train due to the vanishing/exploding gradient problem [30]. Therefore, we adopt the residual connections scheme proposed in [31]. Residual connections directly pass information from the previous layer to the next layer as depicted in Fig. 2.

The output layer of the AUE has dimension equal to the maximum sparsity level $K_{\max}$. The softmax layer takes as input a vector and normalizes it to a probability distribution

$$\hat{p}_i = \frac{e^{z_i}}{\sum_{j=1}^{K_{max}} e^{z_j}} \tag{9}$$

where z_i and z_j are the i^{th} and j^{th} element of $\boldsymbol{z}$, while $\hat{p}_i$ is the i^{th} element of $\hat{\boldsymbol{p}}$. The final estimate is

$$\hat{K} = \arg\max_i \hat{p}_i. \tag{10}$$

For active user identification we use $K_{\max}$ neural networks $g_1(\cdot), g_2(\cdot), \ldots, g_{K_{\max}}(\cdot)$, as defined in (5). The architecture of all the $K_{\max}$ AUI networks remains the same as shown in Fig. 2b, only the dataset used for training each AUI is different. For instance, for training $g_k(\cdot)$, a dataset comprising of k active users is considered. Here, a dropout layer is

Algorithm 1 Deep Learning-Based AUEI

Input: $\tilde{\boldsymbol{y}}$, $K_{\max}$
Output: $\hat{\Omega}$

1: Pass $\tilde{\boldsymbol{y}}$ through the enumeration DNN to obtain $\hat{\boldsymbol{p}}$
2: $\hat{K} \leftarrow \underset{i\in\{1,\ldots,K_{\max}\}}{\arg\max}\ \hat{p}_i$
3: $\hat{\Omega} \leftarrow g_{\hat{K}}(\tilde{\mathbf{y}}, \boldsymbol{\Theta}_{\hat{K}})$

Return: $\hat{\Omega}$

employed to avoid overfitting of the model on the training dataset. In this layer, during the training phase, a fraction of the input and output connections from the neurons are dropped [28]. To identify active users, we adopt as output a sigmoid layer with N outputs, one per user. Each output is calculated as

$$\hat{q}_i = \frac{1}{1+e^{-z_i}} \quad i = 1, \ldots, N \tag{11}$$

where $\hat{q}_i$ represents the likelihood of user i being active. In previous approaches, a comparison with a threshold was proposed to decide which users were active, but these methods suffer from the difficulties in finding a suitable threshold value. In our approach, summarized in Algorithm 1, we rely on $\hat{K}$ from the AUE network and then consider the AUI network trained for $\hat{K}$ active users. With this network, using $\tilde{\boldsymbol{y}}$ as input, the $\hat{K}$ users with the largest likelihoods are considered as active

$$\hat{\Omega} = \underset{\Omega, |\Omega|=\hat{K}}{\arg\max} \sum_{i\in\Omega} \hat{q}_i. \tag{12}$$

B. DNNs TRAINING

Sparsity estimation can be seen as a multi-class classification, in which we are categorizing the input $\tilde{\boldsymbol{y}}$ in one of the categories ranging from 1 to $K_{\max}$. To this aim, we employ a categorical cross-entropy loss. Let us indicate the true label vector as $\boldsymbol{p} = [p_1, p_2, \ldots, p_{K_{\max}}]$. If the number of active users is k, it will be $p_k = 1$ and $p_j = 0\,\forall j \neq k$. For instance, if the number of active users is 2, then $\boldsymbol{p} = [0, 1, 0, \ldots, 0]$. The categorical cross-entropy $\mathcal{J}_S(\boldsymbol{p}, \hat{\boldsymbol{p}})$ loss is defined as

$$\mathcal{J}_S(\boldsymbol{p}, \hat{\boldsymbol{p}}) = -\sum_{i=1}^{K_{\max}} p_i \log \hat{p}_i = -\log \hat{p}_K. \tag{13}$$

User activity identification can be seen as a multi-label classification problem, in which we are selecting $\hat{K}$ out of N users. To this aim, we employ binary cross-entropy loss. Let us indicate the true label vector as $\boldsymbol{q} = [q_1, q_2, \ldots, q_N]$ where each element represents the user as active ($q_i = 1$) or inactive ($q_i = 0$). For instance, if $\Omega = \{2, 4\}$ then $\boldsymbol{q} = [0, 1, 0, 1, \ldots, 0]$. The binary cross-entropy loss is defined as

$$\mathcal{J}_A(\boldsymbol{q}, \hat{\boldsymbol{q}}) = -\sum_{i=1}^{N} (q_i \log \hat{q}_i + (1 - q_i) \log(1 - \hat{q}_i)). \tag{14}$$

In order to determine the parameters Ψ and Θ_k in (5), we need to minimize the loss functions $\mathcal{J}_S(\boldsymbol{p}, \hat{\boldsymbol{p}})$ and $\mathcal{J}_A(\boldsymbol{q}, \hat{\boldsymbol{q}})$ for enumeration and identification, respectively. For that purpose, we employ the well-known Adam optimizer [32].

With the proposed approach, we have to train $K_{\max}$ AUI networks which is a time and computationally expensive task. To counter that, we propose a multi-stage transfer learning technique. To train the AUI network $g_k(\cdot)$ in (5) for $k \geq 2$ through this technique, we start from the weights of $g_{k-1}(\cdot)$. More precisely, the weights of $g_1(\cdot)$ are initialized according to [33]. Then, $g_1(\cdot)$ is trained until the network converges, i.e., there is no significant change in the network weights. Instead of initializing the weights of $g_2(\cdot)$ randomly, they are initialized with the trained weights of $g_1(\cdot)$; this way, $g_2(\cdot)$ leverages the information learnt by $g_1(\cdot)$ and converges faster than its randomly initialized counterpart. In general, the weights of $g_k(\cdot)$ are hence initialized through the trained weights of $g_{k-1}(\cdot)$, for $k = 2, \ldots, K_{\max}$.

C. COMPUTATIONAL COMPLEXITY

In this subsection, the computational complexity of the AUEI is presented in terms of floating point operations (FLOPs). We assume the addition, subtraction, multiplication, division and exponential computation as a single floating point operation, as in [18]. The FLOPs of the convolutional layers are given by

$$C_{\text{conv}_2} = 2 \cdot N_{\text{conv}_2} \cdot F_{\text{conv}_2} \cdot N_d \cdot out_{\text{conv}_2}$$
$$C_{\text{conv}_4} = 2 \cdot N_{\text{conv}_4} \cdot F_{\text{conv}_4} \cdot N_d \cdot out_{\text{conv}_4}$$

where N_{conv_*}, F_{conv_*} and out_{conv_*} represent the number of convolution filters, size of the filter and output shape, respectively. The output of the convolutional layers is fed into a ReLU, having computational complexity

$$C_{\text{ReLU}_2} = N_{\text{conv}_2} \cdot out_{\text{conv}_2}$$
$$C_{\text{ReLU}_4} = N_{\text{conv}_4} \cdot out_{\text{conv}_4}.$$

The number of FLOPs in a fully-connected layer (6) is dictated by the input (in) and output (out) size

$$C_{\text{FC}} = in \cdot out + (in - 1) \cdot out + out.$$

The number of multiplication and addition operations in $\boldsymbol{Wa}$ is given by the term ($in \cdot out$) and ($in \cdot out - out$), respectively. The last term (out) is the number of addition operations due to the bias $\boldsymbol{b}$. The computational complexity of the fully-connected layer simplifies to

$$C_{\text{FC}} = 2 \cdot in \cdot out.$$

Consequently, the FLOPs of the input fully-connected layer can be defined as

$$C_{\text{FC}_{in}} = 2\alpha \cdot [N_{\text{conv}_2} \cdot out_{\text{conv}_2} + N_{\text{conv}_4} \cdot out_{\text{conv}_4}].$$

The batch normalization (7) involves four operations, therefore, the complexity of the input batch normalization layer can be expressed as

$$C_{\text{BN}_{in}} = 4\alpha.$$

The hidden layer is composed of two fully-connected layers, two batch normalization layers, two activation functions, one dropout layer and one residual connection. The dropout layer and residual connection are elementwise multiplication and addition operations; therefore, each will contribute α complexity to the algorithm. The overall complexity of L hidden layers is given by

$$\begin{aligned} C_{\text{hidden}} &= (2\alpha^2 + 2\alpha^2 + 4\alpha + 4\alpha + 2\alpha + \alpha + \alpha)L \\ &= 4\alpha^2 L + 12\alpha L. \end{aligned}$$

The computational cost incurred at the output fully-connected layer of AUE and AUI is

$$C_{\text{FC}_{out}}^{\text{AUE}} = 2\alpha K_{\max}$$

and

$$C_{\text{FC}_{out}}^{\text{AUI}} = 2\alpha N$$

respectively. The softmax layer (9) in AUE invokes $K_{\max}$ exponential, $K_{\max}$ divisions and $K_{\max}-1$ additions operations

$$C_{\text{softmax}} = 3K_{\max} - 1.$$

Similarly, the number of floating point operations in a sigmoid layer (11) is:

$$C_{\text{sigmoid}} = 3N.$$

According to [34], finding the largest probabilities in (10) and (12) yields the following complexity

$$C_{\max}^{\text{AUE}} = K_{\max} - 1$$

and

$$C_{\max}^{\text{AUI}} = KN - \frac{K(K+1)}{2}$$

respectively. The overall computational complexity of the AUE is described below

$$\begin{aligned} C_{\text{AUE}} &= C_{\text{conv}_2} + C_{\text{conv}_4} + C_{\text{ReLU}_2} + C_{\text{ReLU}_4} + C_{\text{FC}_{in}} \\ &\quad + C_{\text{BN}_{in}} + C_{\text{hidden}} + C_{\text{FC}_{out}}^{\text{AUE}} + C_{\text{softmax}} + C_{\max}^{\text{AUE}} \\ &= 2 \cdot (N_{\text{conv}_2} \cdot out_{\text{conv}_2})(F_{\text{conv}_2} \cdot N_d + \alpha) \\ &\quad + 2 \cdot (N_{\text{conv}_4} \cdot out_{\text{conv}_4})(F_{\text{conv}_4} \cdot N_d + \alpha) \\ &\quad + 4\alpha + 4\alpha^2 L + 12\alpha L + 2\alpha K_{\max} + 4K_{\max} - 2. \end{aligned} \tag{15}$$

Likewise, the computational complexity of AUI is given as

$$\begin{aligned} C_{\text{AUI}} &= C_{\text{conv}_2} + C_{\text{conv}_4} + C_{\text{ReLU}_2} + C_{\text{ReLU}_4} + C_{\text{FC}_{in}} \\ &\quad + C_{\text{BN}_{in}} + C_{\text{hidden}} + C_{\text{FC}_{out}}^{\text{AUI}} + C_{\text{sigmoid}} + C_{\max}^{\text{AUI}} \\ &= 2 \cdot (N_{\text{conv}_2} \cdot out_{\text{conv}_2})(F_{\text{conv}_2} \cdot N_d + \alpha) \\ &\quad + 2 \cdot (N_{\text{conv}_4} \cdot out_{\text{conv}_4})(F_{\text{conv}_4} \cdot N_d + \alpha) \\ &\quad + 4\alpha + 4\alpha^2 L + 12\alpha L + 2\alpha N + 3N + KN \\ &\quad - \frac{K(K+1)}{2}. \end{aligned} \tag{16}$$

Finally, the complexity of the AUEI is

$$C_{\text{AUEI}} = C_{\text{AUE}} + C_{\text{AUI}}. \tag{17}$$

In the next section, we compare this complexity with that of the algorithm presented in [18].

IV. IMPLEMENTATION AND RESULTS

A. SIMULATION SETUP

We generate samples according to the system model described by (2) for training and testing our DNNs networks. To compare our proposal with other algorithms from the literature, we choose the same simulation parameters as in [18], namely a total number of users $N = 100$, a maximum number of active users $K_{\max} = 8$, spreading codewords with sparsity $n_S = 2$ and length $S = 10$, and $N_d = 7$ successive measurements. The case of zero active users can be handled with less computationally expensive spectrum sensing techniques or machine learning algorithms, as described, e.g., in [35], [36], and [37]. The non-zero values of the LDS codewords are generated from the distribution $\mathcal{CN}(0,\ \sigma_w^2)$ with $\sigma_w^2 = 1$. We use a Rayleigh fading channel model with perfect power control, so that $h_{i,j} \sim \mathcal{CN}(0,\ 1)$ are i.i.d. complex Gaussian. Note that owing to perfect power control, the distance of the devices from the BS does not contribute towards the received vector. The data symbols s_i are unit energy quadrature phase-shift keying (QPSK), so that the SNR is defined as $\text{SNR} = 1/\sigma_n^2$.

For the AUE network dataset, the number of active users in each sample varies from 1 to $K_{\max}$. For the training, we generate $13.5 \cdot 10^6$ samples. The dataset generation for the k^{th} AUI network $g_k(\cdot)$ involves randomly activating k users from a total of N. We generate $9 \cdot 10^6$ training samples and 10^6 testing samples per AUI network.

The architecture of both the AUE and AUI DNNs consists of $L = 2$ hidden layers. The convolutional layers consist of 64 filters. Except the last fully-connected layer, each fully-connected layers consists of $\alpha = 1000$ neurons. In case of AUE and AUI, the last fully connected layer contains $K_{\max} = 8$ and $N = 100$ neurons, respectively.

We train the sparsity estimation DNN for 10 epochs. Regarding the AUI networks, in order to minimize the training time, we adopt the multi-stage transfer learning approach. Hence, the first AUI network, $g_1(\cdot)$, is trained for 10 epochs with He initialization [33], while for the $g_k(\cdot)$ network the weights are initialized from the trained weights of $g_{k-1}(\cdot)$. We employ the Adam optimizer for learning the weights in both DNN networks. For the optimizer, we consider the following configuration: learning rate $= 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. In the training phase, we consider a mini-batch of size $|\boldsymbol{B}| = 1000$. The drop out rate is set to 0.1.

For the implementation of the deep learning algorithms, we employ Keras deep learning framework with Tensorflow as backend [28], [38]. We trained the DNN algorithms on a GPU server consisting of two Nvidia Quadro RTX 5000 cards, two Intel Xeon Gold 5222 Processors and 128 GB RAM.

B. RESULTS

As for performance metrics, we use the recall defined as $R = \text{TP}/(\text{TP} + \text{FN})$ and the false alarm rate $F = \text{FP}/(\text{FP} + \text{TN})$, where TP, TN, FP, and FN stand for true positive, true

TABLE 1. Recall, $K = 8$, SNR $= 10$ dB.

Epochs	Without TL	With TL
1	0.706	0.736
2	0.728	0.740
3	0.735	0.741
4	0.738	0.742
5	0.739	0.743
6	0.740	0.743
7	0.741	0.744
8	0.741	0.744
9	0.742	0.744
10	0.742	0.744

negative, false positive, and false negative, respectively. True positives (TP) and true negatives (TN) indicate the number of occurrences when the active/inactive users are correctly identified as active/inactive, respectively. Similarly, false positives (FP) and false negatives (FN) represent the number of occurrences when the inactive/active users are misclassified as active/active, respectively. In the following one iteration means updating the weights over a mini-batch.

Let us first investigate the training phase, specifically the rate of convergence of the weights for the AUI networks. In this regard, in Fig. 3 we report the loss versus the number of iterations. Comparing the curves with and without transfer learning, where $g_k(\cdot)$ for $k = 2$, 4 and 8 is trained for 3 epochs for the transfer learning approach, a considerable improvement in the speed of convergence of the training can be observed. The improvement is substantial for all sparsity levels K, and it is particularly important for the networks designed for large K (see, e.g., the case $K = 8$). In the case $K = 2$ the advantage due to transfer learning is less pronounced. We can appreciate the improvement also in terms of recall in Table 1, where we report the results with and without transfer learning for $K = 8$ and SNR $= 10$ dB. For obtaining the recall values through the multi-stage transfer learning, $g_1(\cdot)$ is trained for 10 epochs while $g_k(\cdot)$ for $2 \leq k \leq 8$ are trained for epochs as in the first column of the Table 1. The networks which are trained without the transfer learning approach are initialized through [33].

We compare the recall for the proposed architecture with the points taken from the literature proposing other algorithms, under the same simulation parameters, namely the Deep AUD (D-AUD) [18], and the compressed-sensing Approximate Message Passing (AMP) [18]. The curves for the proposed AUEI are obtained through the multi-stage transfer learning approach. The $g_1(\cdot)$ is trained for 10 epochs while $g_k(\cdot)$ for $2 \leq k \leq 8$ is trained for 3 epochs. The proposed approach shows improved recall values with respect to the other algorithms, as can be seen in Fig. 4 and Fig. 5 for SNR $= 10$ dB and SNR $= 20$ dB, respectively. In contrast to our approach, the other algorithms suffer from substantial performance degradation for high sparsity levels. We present

TABLE 2. False Alarm rate, Transfer Learning, epochs= 3.

Sparsity Level (K)	SNR $= 10$ dB	SNR $= 20$ dB
1	$3.87\text{e}-5$	$1.11\text{e}-7$
2	$2.50\text{e}-4$	$2.05\text{e}-6$
3	$9.46\text{e}-4$	$4.50\text{e}-5$
4	$2.32\text{e}-3$	$2.54\text{e}-4$
5	$4.88\text{e}-3$	$8.10\text{e}-4$
6	$9.00\text{e}-3$	$1.96\text{e}-3$
7	$1.49\text{e}-2$	$4.59\text{e}-3$
8	$1.94\text{e}-2$	$6.24\text{e}-3$

TABLE 3. Computational Complexity in FLOPs, $N_d = 7$.

	$K = 1$	$K = 2$	$K = 4$	$K = 8$
AUEI	$2.02\text{e}+7$	$2.02\text{e}+7$	$2.02\text{e}+7$	$2.02\text{e}+7$
D-AUD	$1.25\text{e}+7$	$2.51\text{e}+7$	$5.01\text{e}+7$	$1.00\text{e}+8$

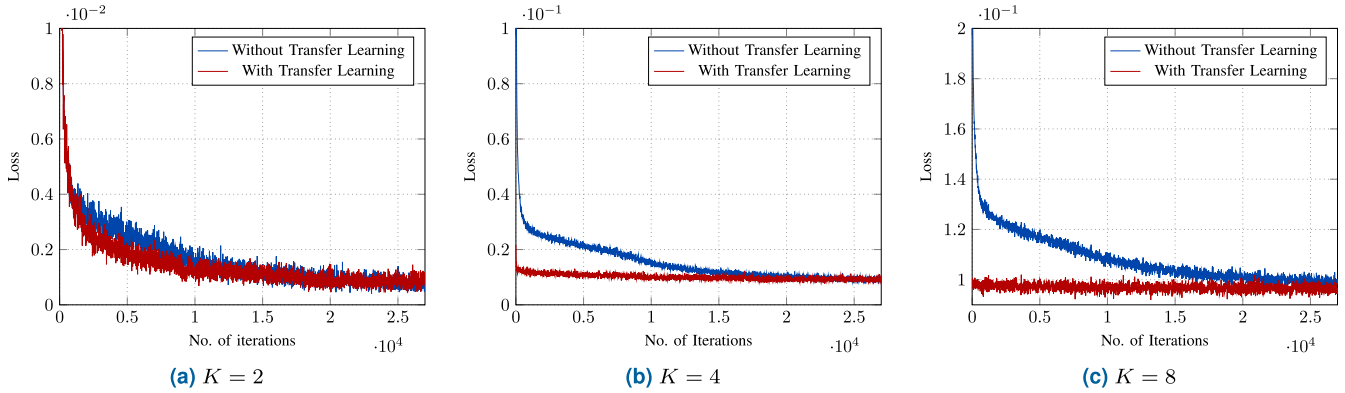
in Table 2 the false alarm rate for the proposed architecture with multi-stage transfer learning. It can be observed that our approach, besides the previously discussed high recall, yields a negligible false alarm rate.

In Fig. 6, we compare the performance of our algorithm with D-AUD and AMP in terms of recall for the SNR range $0 - 20$ dB, $N_d = 7$ and $K = 4$. It can be observed that our approach outperforms the other approaches, especially in the low SNR regime. To check the robustness of our algorithm, we illustrate the performance for overloading factors 125% and 250% in Fig. 7 and 8, respectively. The overloading factor is defined as $N/(N_d S)$. For different overloading factors, we assume a fixed length of the spreading sequence, S, and a number of users, N, while varying the number of measurements, N_d. A significant performance improvement can be observed for $N_d = 8$ in comparison to $N_d = 4$ for all the algorithms. In other words, increasing the number of measurements N_d or reducing the overloading factor yields better performance. We observe that the proposed algorithm outperforms the D-AUD and AMP in both scenarios, confirming the reliability of AUEI.

Finally, we present the numerical comparison of computational complexity between AUEI (see Section III-C) and D-AUD, whose complexity for a given sparsity K is stated in [18] as

$$C'_{\text{D-AUD}} = 2L\alpha^2 + (4N_dS + 7L + 2N + 4)\alpha + (K+3)N - \frac{K(K+1)}{2} - 1.$$

For calculating the overall D-AUD complexity, we also take into account the algorithm proposed in [18] for sparsity estimation. In this algorithm, the received vector is passed first through the D-AUD trained for sparsity level $K = 1$. If the output satisfies the threshold-based condition, this is considered as the sparsity level. Otherwise, the received vector is



(a) $K = 2$ (b) $K = 4$ (c) $K = 8$

FIGURE 3. Training Loss with Transfer Learning and without Transfer Learning, SNR = 10 dB.

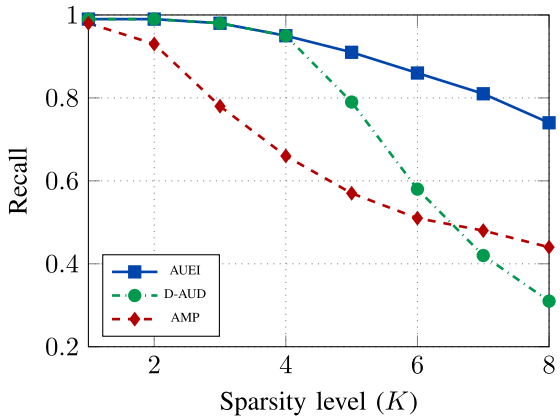


FIGURE 4. Recall vs. sparsity level, SNR = 10 dB.

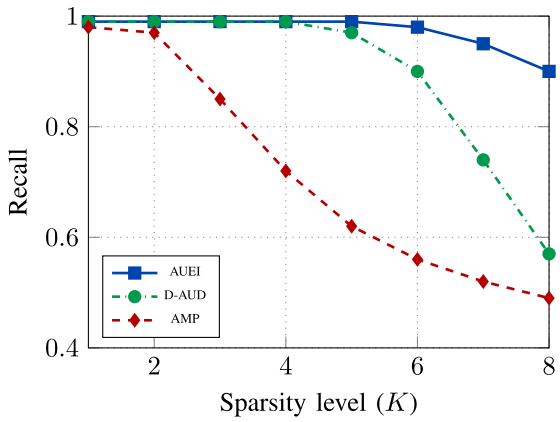


FIGURE 5. Recall vs. sparsity level, SNR = 20 dB.

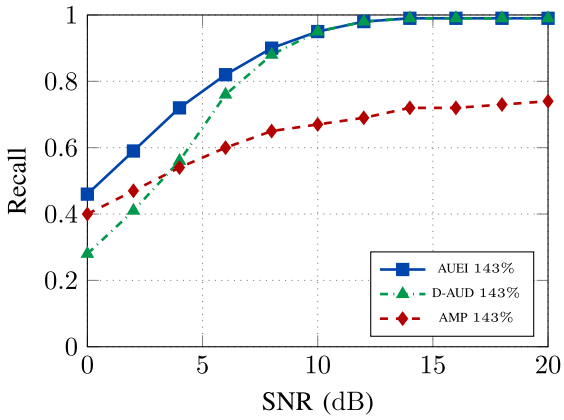


FIGURE 6. Recall vs. SNR, $N_d = 7, K = 4$.

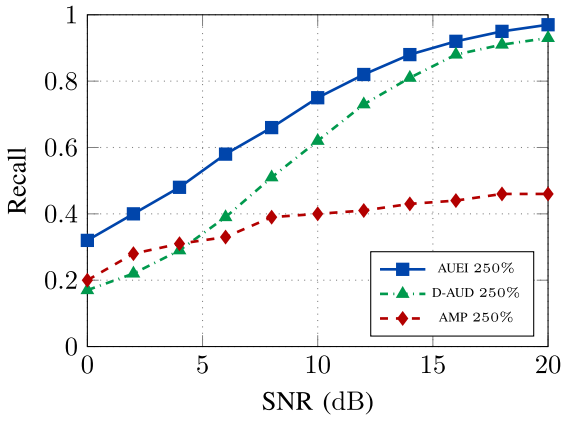


FIGURE 7. Recall vs. SNR, $N_d = 4, K = 4$.

passed through the D-AUD network trained for $K = 2$, and so on. The procedure is repeated until the threshold-based condition is met or the maximum sparsity level is reached. Thus, for a given sparsity K, the received vector is passed through K D-AUDs. For this reason, the complexity of the D-AUD algorithm grows linearly with the sparsity level. Considering that, the overall computational complexity expression for D-AUD is

$$C_{\text{D-AUD}} = KC'_{\text{D-AUD}}. \tag{18}$$

Table 3 shows the computational complexity of AUEI and D-AUD for $N_d = 7$ and $K = 1, 2, 4$, and 8, calculated through (17) and (18). The number of hidden layers for AUEI and D-AUD is $L = 2$ and $L = 6$, respectively. As observed, the computational complexity of D-AUD increases

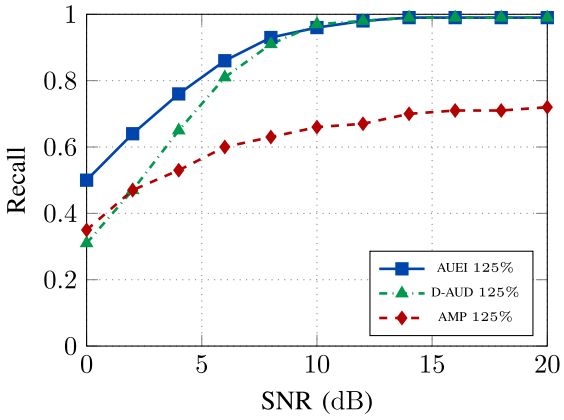


FIGURE 8. Recall vs. SNR, $N_d = 8, K = 4$.

linearly with the sparsity level, while the complexity of AUEI remains practically constant. This is due to the fact that the dependence on the sparsity level K in (16) has a negligible effect on the overall computational complexity. Specifically, for all cases with more than one active user, the AUEI shows a significant gain in terms of complexity. So, despite having two separate architectures instead of one as in D-AUD, our approach yields a lower complexity and better performance.

V. CONCLUSION

In this paper, we have proposed an active users detection method, realized by one DNN for active users enumeration and one for active users identification. We designed the deep neural network architectures to extract relevant features from the multiple measurements for enumeration and identification. Besides the fully-connected layers, both DNNs consist of convolutional layers to reduce the computational complexity. To minimize the training time for the active users identification networks, we adopted the multi-stage transfer learning technique. The numerical results demonstrate that our approach is more effective than previously known methods in identifying the active users, especially for high sparsity levels and low SNR. We also analyzed the false alarm rates, which are negligible for the scenarios of interest, and the computational complexity, which results lower than other approaches.

Future work will include analysis of the scalability of the proposed algorithm for a different number of users and further reduction of the computational cost.

ACKNOWLEDGMENT

This work has been carried out in the framework of the CNIT National Laboratory WiLab and the WiLab-Huawei Joint Innovation Center.

REFERENCES

[1] C. Bockelmann, N. K. Pratas, G. Wunder, S. Saur, M. Navarro, and D. Gregoratti, "Towards massive connectivity support for scalable mMTC communications in 5G networks," *IEEE Access*, vol. 6, pp. 28969–28992, 2018.

[2] F. Li, D. Wang, Y. Wang, X. Yu, N. Wu, J. Yu, and H. Zhou, "Wireless communications and mobile computing blockchain-based trust management in distributed Internet of Things," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–12, Dec. 2020.

[3] Y. Ahn, W. Kim, and B. Shim, "Active user detection and channel estimation for massive machine-type communication: Deep learning approach," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 11904–11917, Jul. 2022.

[4] M. Ke, Z. Gao, Y. Wu, X. Gao, and R. Schober, "Compressive sensing-based adaptive active user detection and channel estimation: Massive access meets massive MIMO," *IEEE Trans. Signal Process.*, vol. 68, pp. 764–779, 2020.

[5] C. Bockelmann, H. F. Schepker, and A. Dekorsy, "Compressive sensing based multi-user detection for machine-to-machine communication," *Trans. Emerg. Telecommun. Technol.*, vol. 24, no. 4, pp. 389–400, Jun. 2013.

[6] 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.

[7] S. Haghighatshoar, P. Jung, and G. Caire, "Improved scaling law for activity detection in massive MIMO systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 381–385.

[8] T. Hara, H. Iimori, and K. Ishibashi, "Hyperparameter-free receiver for grant-free NOMA systems with MIMO-OFDM," *IEEE Wireless Commun. Lett.*, vol. 10, no. 4, pp. 810–814, Apr. 2021.

[9] Y. Bai, B. Ai, and W. Chen, "Deep learning based fast multiuser detection for massive machine-type communication," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–5.

[10] R. Garg and R. Khandekar, "Block-sparse solutions using kernel block rip and its application to group lasso," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, vol. 15, Apr. 2011, pp. 296–304.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Dec. 1989.

[12] G. Hinton, L. Deng, D. Yu, and G. E. Dahl, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-V4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 4278–4284.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Dec. 2014, pp. 3104–3112.

[15] Z. Qin, H. Ye, G. Y. Li, and B. H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 93–99, Mar. 2019.

[16] T. Wang, C. K. Wen, H. Wang, and F. Gao, "Deep learning for wireless physical layer: Opportunities and challenges," *China Commun.*, vol. 14, no. 11, pp. 92–111, Nov. 2017.

[17] L. Dai, R. Jiao, F. Adachi, H. V. Poor, and L. Hanzo, "Deep learning for wireless communications: An emerging interdisciplinary paradigm," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 133–139, Aug. 2020.

[18] W. Kim, Y. Ahn, and B. Shim, "Deep neural network-based active user detection for grant-free NOMA systems," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2143–2155, Apr. 2020.

[19] J. Wu, W. Kim, and B. Shim, "Pilot-less one-shot sparse coding for short packet-based machine-type communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9117–9120, Aug. 2020.

[20] L. Wei, S. Lu, H. Kamabe, and J. Cheng, "User identification and channel estimation by DNN-based decoder on multiple-access channel," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[21] J. H. I. de Souza and T. Abrao, "Deep learning-based activity detection for grant-free random access," *IEEE Syst. J.*, early access, Jun. 6, 2022, doi: 10.1109/JSYST.2022.3175658.

[22] R. Hoshyar, F. P. Wathan, and R. Tafazolli, "Novel low-density signature for synchronous CDMA systems over AWGN channel," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1616–1626, Apr. 2008.

[23] Y. Fu, H. Li, Q. Zhang, and J. Zou, "Block-sparse recovery via redundant block OMP," *Signal Process.*, vol. 97, pp. 162–171, Apr. 2014.

[24] A. Elzanaty, A. Giorgetti, and M. Chiani, "Weak RIC analysis of finite Gaussian matrices for joint sparse recovery," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1473–1477, Oct. 2017.

[25] A. Elzanaty, A. Giorgetti, and M. Chiani, "Limits on sparse data acquisition: RIC analysis of finite Gaussian matrices," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1578–1588, Mar. 2019.

[26] T. Sivalingam, S. Ali, N. H. Mahmood, N. Rajatheva, and M. Latva-Aho, "Deep neural network-based blind multiple user detection for grant-free multi-user shared access," 2021, *arXiv:2106.11204*.
[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org
[28] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras
[29] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with ReLU activation function and linear spline-type methods," *Neural Netw.*, vol. 110, pp. 232–242, Feb. 2019.
[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[34] J. Wang, S. Kwon, and B. Shim, "Generalized orthogonal matching pursuit," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6202–6216, Dec. 2012.
[35] A. Mariani, S. Kandeepan, and A. Giorgetti, "Periodic spectrum sensing with non-continuous primary user transmissions," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1636–1649, Mar. 2015.
[36] E. Recayte, A. Munari, and F. Clazzer, "Grant-free access: Machine learning for detection of short packets," in *Proc. 10th Adv. Satell. Multimedia Syst. Conf. 16th Signal Process. Space Commun. Workshop (ASMS/SPSC)*, Oct. 2020, pp. 1–7.
[37] E. Testi and A. Giorgetti, "Blind wireless network topology inference," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1109–1120, Feb. 2021.
[38] M. Abadi. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

MUHAMMAD USMAN KHAN (Graduate Student Member, IEEE) received the B.S. degree *(cum laude)* in electrical engineering from the National University of Computer and Emerging Sciences, Lahore, Pakistan, in 2016, and the M.Sc. degree (Hons.) in advanced electronic systems engineering from the University of Kent, Canterbury, U.K., in 2018. He is currently pursuing the Ph.D. degree in electronics, telecommunications, and information technologies with the University of Bologna, Cesena, Italy. His research interests include active users detection and channel estimation in the massive Internet of Things (mIoT) scenario using deep learning. He worked as a Full-Time Faculty Member at the School of Computing, National University of Computer and Emerging Sciences, Lahore, Pakistan. He also worked as a mentor, student ambassador, and course representative from 2017 to 2018 at the University of Kent.

ENRICO PAOLINI (Senior Member, IEEE) received the Dr.Ing. degree *(summa cum laude)* in telecommunications engineering and the Ph.D. degree in electrical engineering from the University of Bologna, Italy, in 2003 and 2007, respectively.

While working toward the Ph.D. degree, he was a Visiting Research Scholar with the Department of Electrical Engineering, University of Hawai'i, Manoa, Honolulu, HI, USA. He was a Visiting Scientist with the Institute of Communications and Navigation, German Aerospace Center, in 2012 and 2014, under DLR-DAAD fellowships. He is currently an Associate Professor with the Department of Electrical, Electronic, and Information Engineering, University of Bologna. His research interests include digital communication systems, error-correcting codes, massive multiple access protocols, and detection and tracking in radar systems. He served as Co-Chair for the ICC 2014, ICC 2015, and ICC 2016 Workshop on Massive Uncoordinated Access Protocols (MASSAP), the VTC 2019-Fall Workshop on Small Data Networks, the 2018 IEEE European School of Information Theory (ESIT), and the 2020 IEEE Information Theory Workshop (ITW 2020). He served as TPC Co-Chair for the IEEE GLOBECOM 2022–Communication Theory Symposium and the IEEE GLOBECOM 2019–Communication Theory Symposium. He is the Chair of the ITSoc Italy Section Chapter and the Secretary of the IEEE ComSoc Radio Communications Committee. He was an Editor of IEEE COMMUNICATIONS LETTERS from 2012 to 2015 and the IEEE TRANSACTIONS ON COMMUNICATIONS (in coding and information theory) from 2015 to 2020.

MARCO CHIANI (Fellow, IEEE) received the Dr.Ing. degree *(summa cum laude)* in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Bologna, Italy, in 1989 and 1993, respectively. He is currently a Full Professor of telecommunications with the University of Bologna. Since 2003, he has been a frequent visitor at the Massachusetts Institute of Technology (MIT), Cambridge, where he presently holds a research affiliate appointment. His research interests include information theory, wireless systems, statistical signal processing, and quantum information. He received the 2011 IEEE Communications Society Leonard G. Abraham Prize in the field of communications systems, the 2012 IEEE Communications Society Fred W. Ellersick Prize, and the 2012 IEEE Communications Society Stephen O. Rice Prize in the field of communications theory. He served as an Elected Chair for the Radio Communications Committee of the IEEE Communication Society from 2002 to 2004 and as an Editor for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2000 to 2007.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement