Wireless Personal Communications (2021) 118:281–299
https://doi.org/10.1007/s11277-020-08013-7

A Deep Neural Network Model for Hybrid Spectrum Sensing in Cognitive Radio

A. Nasser[1,2] · M. Chaitou[4] · A. Mansour[1] · K. C. Yao[3] · H. Charara[4]

Accepted: 27 November 2020 / Published online: 3 January 2021

Abstract
Spectrum sensing (SS) is an essential task of the secondary user (SU) in a cognitive radio system. SS monitors the primary user (PU) activity in order to avoid any collision with SU, as the latter should be silent when PU is active on a given channel. Hybrid SS (HSS) is one of the powerful methods used to monitor PU activity. It consists of using different detectors together to make a final decision on the PU status. In this manuscript, artificial neural networks (ANN) are used to perform HSS. Since our data is composed from the test statistics (TSs) of several detectors, thus it can be modeled as tabular. Fully connected neural networks become the most suitable ANN model. We applied cutting-edge techniques in the field of deep learning in order to get the best possible accurate neural network model in our application. These techniques boil down to: embedding, regularization, batch normalization and smart learning rate selection. With the help TSs related to several detectors, ANN is trained to distinguish between two hypotheses, H_0: PU is absent and H_1: PU is active. Numerical results show the effectiveness of our proposed ANN-based HSS, as it outperforms the classical ANN-based energy detector and proves its capability to detect PU signal at very low SNR.

Keywords Spectrum sensing · Artificial neural network · Cognitive radio

1 Introduction

Cognitive Radio has been proposed in order to overcome the spectrum scarcity problem. Unlicensed, namely known as Secondary User (SU) may opportunistically access the channel of the licensed user known as primary user (PU) when the latter is absent [1]. Thus, one of the most important functions in CR becomes the spectrum sensing (SS), which

✉ A. Nasser
abbass.nasser@ensta-bretagne.org

1 LABSTICC, CNRS, UMR 6285, ENSTA Bretagne, 2 Rue François Verny, 29806 Brest, France

2 ICCS-Lab, Computer Science Department, American University of Culture and Education, Beirut, Lebanon

3 LABSTICC, CNRS, UMR 6285, UBO, 6 Avenue le Gorgeu, 29238 Brest, France

4 Computer Science department, Faculty of Science, Lebanese University, Nabatieh, Lebanon

is responsible to verify the primary channel status whether it is occupied or not. Several detector have been proposed to perform the SS tasks, such as: energy detector (ED), auto-correlation detector (ACD) and cyclo-stationary detector (CSD) [2].

In classical SS, i.e. signal detection, the SU applies a test statistic (TS) on the received signal and compares it to a predefined threshold in order to make a decision on the PU status. If the TS is above a certain threshold, then PU is considered as active. In fact, in order to set the optimal threshold that meets the target detection and false alarm rates, this approach predetermines that the statistical distribution of TS is known, which is not always possible due to the unstable, and may be unknown, statistical properties of the noise, the PU signal or the transmission channel.

To overcome the analytic statistical problems of the classical SS and improve its performance, several published works propose the adoption of the machine learning (ML) and the neural networks (NN) techniques in order to make decisions on the PU channel occupancy [3–9]. The main aim of the proposed works is to tune ML or NN systems with the statistics of both hypotheses: the first one is H_0 when PU is assumed to be absent, and H_1 when PU is assumed to be active.

In [5], ML techniques such as the K-means and support-vector machine (SVM) are used to distinguish between the H_0 and H_1 hypotheses in a cooperative SS. Two low-dimension probability vectors related to both H_0 and H_1 of ED are used in order to train the system. SVM is used in order to set the threshold curve between H_0 and H_1 clusters. K-nearest-based ML is adopted in [10] for a cooperative SS. The related mechanism of the proposed work is divided into two phases: training and classification. The global decision of the presence/absence taken at the end of the classification phase of the PU takes into consideration the reliability of each CR user when reporting to the fusion center during the training phase.

For a local SS, an ensemble classifier is proposed in [11]. The classifier seeks to discriminate between H_0 and H_1 hypotheses by being trained with the extracted cyclic features of PU's signal in low SNR conditions. This ensemble classifier is based on decision trees and AdaBoost algorithm. Wideband SS is tackled in [12], where three ML techniques: neural networks, expectation maximization and k-means are used in order to detect presence of one or multiple primary users in a wideband spectrum.

In order to enhance the accuracy of the ML system in making decision on the PU status, hybrid SS (HSS) has been proposed [6, 7]. HSS consists of making a sensing decision based on several detectors instead of considering only one as per the classical SS. In [6, 7], Artificial neural network (ANN) have been applied in order to perform a HSS. ANN is trained using the TSs of two detectors related to H_0 and H_1 (in [6] ED and cyclostationary detector (CSD) are used and in [7] ED and likelihood ratio statistics are used).

The strength of the HSS consists on compensate the weak points of a given detector by the advantages of the another one. For instance, ED suffers from the noise uncertainty at low SNR, which is overcome by ACD. In return, ACD is adversely impacted by the low oversampling rate of the PU signal, while ED is not affected by this issue. A HSS scheme is proposed in [13], where ED and CSD are adopted. First, ED is evaluated to verify whether primary user is present or not. The CSD is used when energy detector is not sure about the presence or absence of PU. Moghimi et al. [14] and Cardenas-Juarez et al. [15] exploit the ED and the waveform detector (WFD) which is coherent detector that is based on the correlation of the received PU signal with a known reference of this signal. An optimal hybrid detector based on ED and WFD is derived as a linear combination of an energy detection metric and a coherent correlation metric.

However, the classical dealing with the HSS requires the knowledge of some statistical features of the combined detectors. This may be hard to obtain since the PU signal's statistical parameters are not always known/available. This fact makes the numerical techniques such as NN an efficient solution. In return, even when NN was used in literature, the hybridization was limited to two detectors as in [6, 7], which does not reflect the real potential of such technique.

In this paper, we present a more general study on the performance of the HSS by admitting up to six different detectors. ANN are trained by the TSs of the detectors using data related to H_0 and H_1. A discussion on the performance is presented according to several criterion related to the ANN itself and the number of detectors to be combined in HSS. Regarding the ANN system, a discussion on the number of layers and the number of nodes in each layer is detailed showing the effect of them on the accuracy of the decision on the PU channel status. For the adopted detectors, the performance is evaluated based on the Probability of Detection, *PD*, and the False Alarm Rate (FAR). In addition, the impact of the number of combined detectors in HSS on the performance is detailed.

The remaining of this paper is organized as follows. In Sect. 2, our system model on the PU signal and the noise is presented. The data model, the neural network model, and the discrimination process between the two hypotheses H_0 and H_1 are given in Sect. 3. Numerical results and discussions are provided in Sect. 4. Finally Sect. 5 concludes our work.

2 System Model

The decision in SS is binary where two hypotheses must be distinguished H_0 and H_1:

$$\begin{cases} H_0 : \text{PU is absent} \\ H_1 : \text{PU is active} \end{cases} \tag{1}$$

The measured TS value leads SU to decide on the PU activity by comparing TS to a predefined threshold.

Accordingly, two classes of TS values have to be defined: H_0-class and H_1-class related to the hypotheses H_0 and H_1 respectively. In fact, H_0-class only depends on the system parameters such as the noise and the hardware imperfections, in other words it is independent from the PU signal since the received signal $r(n)$ can be presented as follows:

$$\begin{cases} r(n) &= w(n) \text{ under } H_0 \\ r(n) &= s(n) + w(n) \text{ under } H_1 \end{cases} \tag{2}$$

where $w(n)$ stands for an additive white Gaussian noise (AWGN) and $s(n)$ is assumed to be the received PU signal to be detected.

For HSS, the SU evaluates a $m \times 1$-dimension vector V related to m detectors.

$$V = [T_{D_1}, T_{D_2}, \ldots, T_{D_m}]^{tr} \tag{3}$$

where the upper script tr stands for the transpose operation, T_{D_i} is the TS related to the detector D_i, $i \in [1, m]$. Each TS is a mathematical application applied on $r(n)$. For instance, ED evaluates the sum of squares of the samples of $r(n)$, whereas ACD stands for the correlation between $r(n)$ and a shifted version of itself, and so on. In classical SS, SU may evaluate only one TS related to a given detector. This TS is compared to a threshold to take

	acd	cpsd	ed	evm	evmm	gof	snr	label
count	9.000000e+06	9.000000e+06	9.000000e+06	9.000000e+06	9.000000e+06	9.000000e+06	9.000000e+06	9000000.0
mean	8.360484e-02	4.549187e+01	1.111497e+00	1.214156e+00	1.184211e+00	3.273784e+01	-1.200000e+01	0.5
std	1.888565e-01	1.023925e+02	2.518895e-01	4.311794e-01	2.946819e-01	4.057269e+00	7.745967e+00	0.5
min	-1.107762e-01	-4.510299e+01	8.461729e-01	8.636874e-01	1.000014e+00	2.518911e+01	-2.400000e+01	0.0
25%	-6.110891e-03	-2.306111e+00	9.909367e-01	1.017553e+00	1.042600e+00	3.025319e+01	-1.800000e+01	0.0
50%	1.448009e-02	6.478896e+00	1.019929e+00	1.050792e+00	1.070963e+00	3.159419e+01	-1.200000e+01	0.5
75%	5.554125e-02	2.788516e+01	1.076245e+00	1.127850e+00	1.132080e+00	3.359676e+01	-6.000000e+00	1.0
max	9.712513e-01	5.107984e+02	2.260447e+00	3.185111e+00	2.728242e+00	6.286738e+01	0.000000e+00	1.0

Fig. 1 Dataset description: 9×10^6 rows; 7 features (6 detectors and SNR); the label has two possible values: 0 for hypothesis H_0 and 1 for hypothesis H_1. The mean, min, max, standard deviation and percentiles (25%, 50% and 75%) of the features and the label are also presented

a decision on PU status. However, in HSS, a vector of TSs related to several detectors are evaluated and combined in order to examine the PU channel status. In our work, this ANN is used to combine the data of these detectors and exploit them in outcome a final decision on PU.

3 The Data Model

In this section we present the details about our dataset and the ANN model used in order to combine the evaluated TSs of the adopted detectors. By training the ANN system with hybrid data, we use such system to make a decision on the PU status.

3.1 Dataset

The data consists of two categories according to the two hypotheses H_0 and H_1. The data was generated corresponding to the TSs of six detectors: ED [16], ACD [17], maximum eigenvalue detector (EVM) [18], maximum–minimum eigenvalue detector (EVMM) [18], cumulative power spectral density detector (CPSD)[19] and goodness-of-fit detector (GoF) [20]. The data respects an AWGN noise and a 16-QAM modulated PU signal with an oversampling rate $N_s = 4$. The TSs related to the adopted detectors are given in the "Appendix". Our dataset, as depicted by Fig. 1, consists of seven features which are {ED, ACD, EVM, EVMM, CPSD, GoF, SNR} and a label. The label values are 0 under hypothesis H_0 and 1 under H_1. Figure 1 presents a description of the dataset. In particular, the dataset contains 9×10^6 rows. Our choice to include the SNR into the set of features is an important issue. Indeed, this prevents building a separate neural network model (NN model) and from training it over each SNR value.

We splitted the dataset randomly into 80% training set and 20% validation set. Figure 2 illustrates the count of rows with H_0 and H_1 respectively (i.e. labels 0 and 1) in the validation set. It can be observed that the data is uniformly distributed among all SNR values. This also applies to the training set.

In order to carefully analyse the data may look in depth, we picked out 1000 random samples from the validation dataset and we plot the scattering of two detectors: ED and ACD as depicted in Fig. 3. The H_1 data drifts away from the H_0 data class as the SNR

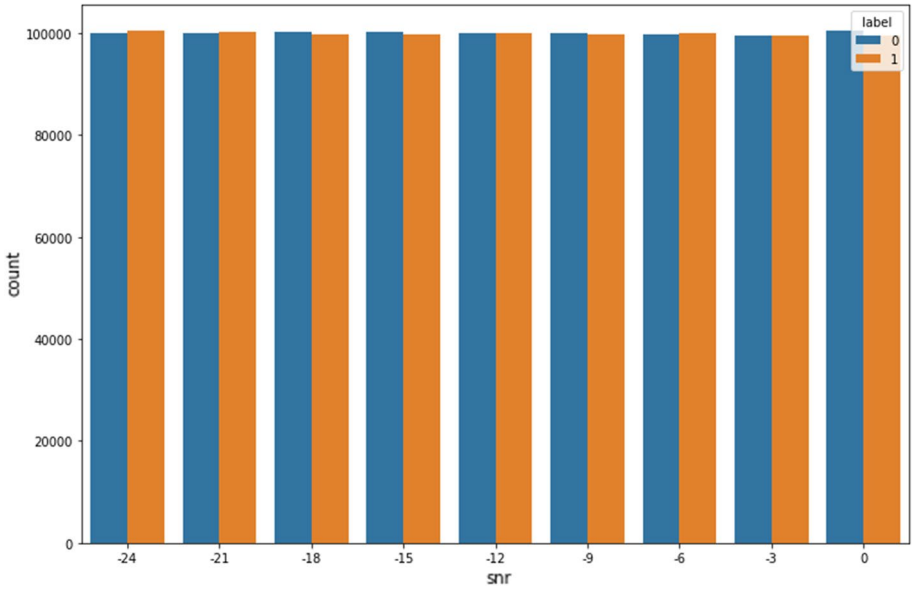


Fig. 2 Histogram of Dataset to show the distribution of the data over the hypotheses H_0 and H_1. These two hypotheses are uniformly considered in our simulations with respect to various SNR values

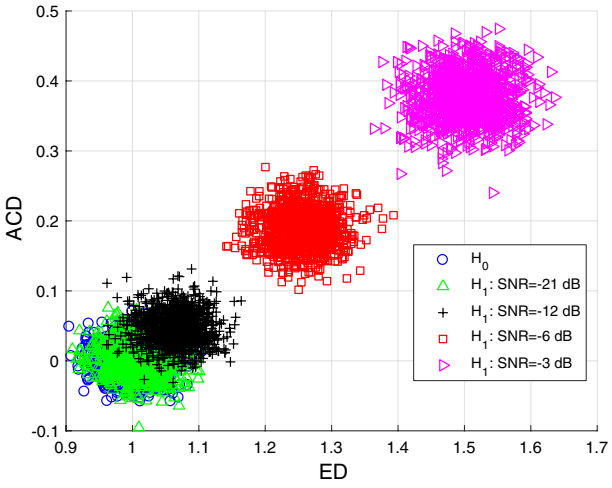


Fig. 3 The scattering of (ξ, α) for N = 1500 samples, 10,000 trials and different values of SNR

increases. H_0 data keeps the same place in the space of the scattering for all SNR values because it is only related to the noise. However, low SNR values (i.e. − 21 dB) makes the discrimination between H_0 and H_1 a tough task due to the huge mix-up of H_0 and H_1 related data (see Fig. 3). However, at a relatively good SNR value (i.e. 6 dB), the classification becomes an easy task.

The data input of the model is a batch of 64 rows (see Sect. 4.1 for a discussion on the batch size). Figure 4 illustrates the first 10 rows of a batch drawn randomly from the dataset.

Fig. 4 10 rows from a batch

snr	acd	cpsd	ed	evm	evmm	gof	target
-24	-0.3629	-0.4049	-0.4387	-0.4545	-0.5065	-1.2119	0
-21	-0.5524	-0.5027	-0.3406	-0.4619	-0.4698	-0.6729	1
-9	0.0251	0.0011	0.1566	-0.0304	-0.0107	0.4884	1
-15	-0.3725	-0.5050	-0.5673	-0.5277	-0.4964	-0.5405	0
-21	-0.4363	-0.4060	-0.2420	-0.4034	-0.4529	-0.0191	1
-24	-0.3593	-0.4638	-0.5214	-0.4976	-0.4878	-0.2024	0
-9	0.1168	0.0626	-0.0245	0.0278	0.0947	0.0138	1
-18	-0.3806	-0.3651	-0.3465	-0.3570	-0.3780	-0.0945	0
-15	-0.3882	-0.4051	-0.2034	-0.4594	-0.5490	-0.3565	1
-12	-0.4552	-0.4373	-0.4157	-0.4741	-0.6051	-0.7295	0

We iterate on the training set by selecting a batch on each step and we fed it as an input to Algorithm 1. After completing a whole pass on the training set, we switch to the validation set and we apply Algorithm 2 in order to assess the accuracy of the model. This completes one epoch. This procedure can be repeated until getting an acceptable value of the accuracy (e.g. an accuracy value $> 95\%$).

3.2 The Neural Network Model

Since our data is in tabular form, we select a fully connected neural network (FCNN). A FCNN consists of one input layer, several hidden layers and one ouput layer. The features' set is the input layer for our model. The output layer will simply consist of two nodes because we are trying to predict whether a row of features' values belongs hypothesis H_0 or H_1. That is, the values of the two output nodes will be two probability values that sum to one. It remains to specify the number of hidden layers, i.e. the ones between the input and output layers,

The number of hidden layers and the number of nodes in each layer, are considered as hyper-parameters and can be tweaked. Two layers are considered. The first layer with 1000 nodes and the second one with 500 nodes. We give a discussion of the model parameters' tweaking in Sect. 4.1.

As a subtle point, notice that the SNR has discrete values, hence it is considered as a categorical variable as opposite to the six detector variables which are continuous. It is a common behaviour to use embedding [21] in the case of a categorical variable since it leads to improve the model accuracy. The embedding process is shown in Fig. 5. In this figure, we take a one-hot encoded vector [21] of SNR concatenated with a bias (i.e. a real value which will be learnt by the NN) which yields a vector of length 10. This vector is mapped to a vector of length 6, called the embedding vector. The embedding vector dimension is a hyper-parameter and can be tweaked (Sect. 4.1). A bias is added because this is required by the embedding process. We concatenate this $6 - D$ vector with the six detectors (Eq. 3) in order to produce the input layer of the FCNN (Fig. 5). Then, we add two hidden layers with [1000, 500] nodes and an output layer with 2 nodes.

For the performance metrics, we select the binary negative log likelihood (NLL) loss function [22] because the type of our problem is binary classification.

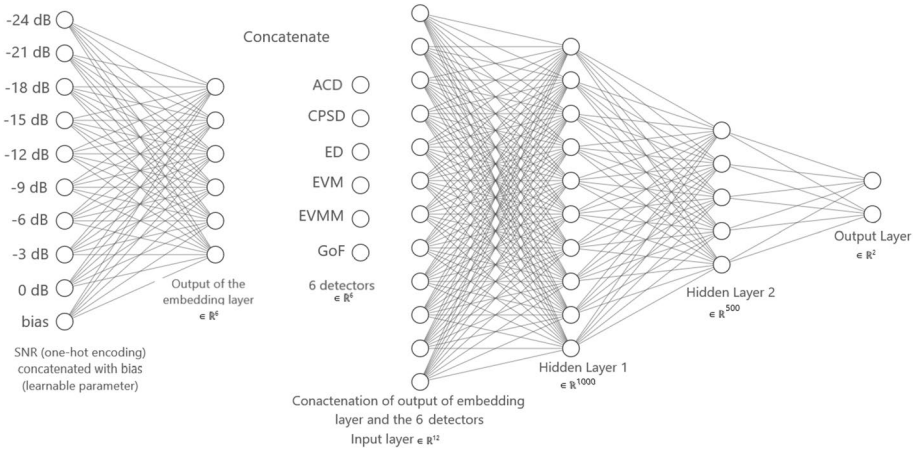


Fig. 5 The NN model archirecture: On the left we see the embedding layer. The output of the embedding layer, which is a vector of 6 real values, is concatenated to the vector of 6 detectors (6 real values) in order to produce the input layer of the NN (a vector of 12 real values). Then we add sequentially: Hidden Layer 1 (a vector of 1000 real values), Hidden Layer 2 (a vector of 500 real values) and the output layer (a vector of 2 real values)

A brief explanation of the NLL loss function is given hereinafter: Let us take a features' row from the dataset. The ground truth label (or target) of this row is 0 or 1 (e.g. the first row in Fig. 4, has a ground truth label = 0). After SNR embedding and concatenation with the other features as explained before, we get a vector x of dimension (12, 1) (the input layer in Fig. 5). Call the output layer $\hat{y} = [\hat{y_0}, \hat{y_1}]^{tr}$ where $\hat{y}_i, i = 0, 1$ is the probability of getting $H_i, i = 0, 1$ as prediction and the upperscript tr is the transpose operator. We encode the ground truth label using one-hot encoding [21]. That is, label 0 is encoded as vector $y = [1, 0]^{\mathrm{T}}$ whereas label 1 is encoded as $y = [0, 1]^{tr}$. That is $y = [y_0, y_1]$ where $y_0 = 1$ if label = 0 and $y_0 = 0$ if label =1. Note that, $y_1 = 1 - y_0$. The binary NLL loss function for this row (e.g. row 1) is expressed as:

$$L_1 = -y_0 \log(\hat{y_0}) - (1 - y_0) \log(1 - \hat{y_0})$$

For a batch of 64 rows, the loss function becomes:

$$L = \sum_{n=1}^{64} -y_{n0} \log(\hat{y}_{n0}) - (1 - y_{n0}) \log(1 - \hat{y}_{n0})/64 \tag{4}$$

where y_{n0} (resp. $\hat{y}_{n0}$) is the encoded label value (resp. predicted probability) of row n of the batch. During the training phase (Algorithm 1) the model will try to minimize the loss function. During the validation phase (Algorithm 2), the loss is also calculated. In addition, we get the confusion matrix and we will derive from it the model accuracy. Furthermore, we well obtain two other important metrics which are the detection probability and the false alarm rate (these two also are derived from the confusion matrix). An example is given in Fig. 6 where the True Positive $TP = 898{,}399$, the False Positive $FP = 62{,}934$,the False Negative $FN = 2246$ and the True Negative $TN = 836{,}421$. Hence, we get the accuracy as :$\frac{TP+TN}{P+N} = 0.9637$ ($P + N$ is the total count of the validation set which is 1800, 000).

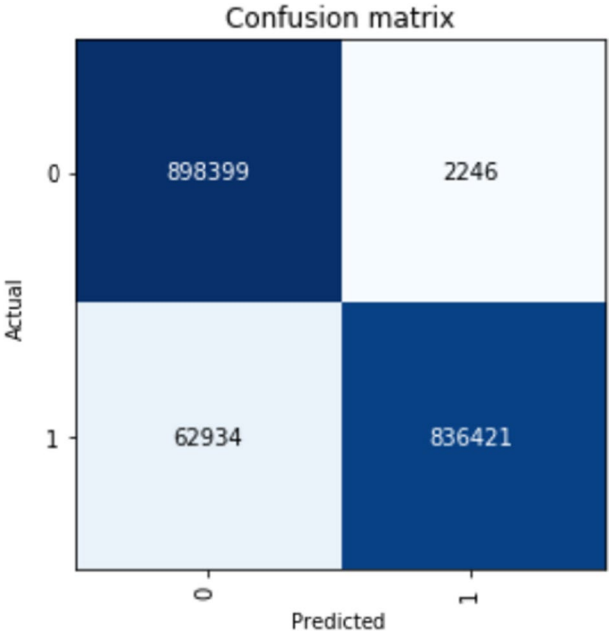


Fig. 6 An example of the confusion matrix on the validation set showing the actual and predicted values, where 0 (resp. 1) represents H_0 (resp. H_1)

```
TabularModel(
  (embeds): ModuleList(
    (0): Embedding(10, 6)
  )
  (emb_drop): Dropout(p=0.0)
  (bn_cont): BatchNorm1d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (layers): Sequential(
    (0): Linear(in_features=12, out_features=1000, bias=True)
    (1): ReLU(inplace)
    (2): BatchNorm1d(1000, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.001)
    (4): Linear(in_features=1000, out_features=500, bias=True)
    (5): ReLU(inplace)
    (6): BatchNorm1d(500, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): Dropout(p=0.01)
    (8): Linear(in_features=500, out_features=2, bias=True)
  )
)
```

Fig. 7 The NN model details: we begin by an embedding layer transforming a list of 10 values [i.e. 9 SNR values and a bias (see Fig. 5)] to a vector of 6 real values. Then we apply batch normalisation to it and we concatenate it with the 6 detectors' values. Then we apply sequentially two hidden layers and on each layer we apply ReLU, batch normalisation and dropout. Finally, we add the output layer

Consequently, the Detection Probability PD can be evaluated as: $PD = \frac{TP}{TP+FP} = 0.9345$ and the False Alarm Rate FAR is: $FAR = \frac{FN}{FN+TN} = 0.002678$.

The details of the model are described in Fig. 7. First, an embedding layer is constructed as discussed before. Then, we apply a regularization technique called Dropout[1] [23]. Dropout consists of dropping a percentage of a layer nodes randomly in the training process. This percentage is determined by the value p in Fig. 6. For the embedding layer, we put $p = 0$, that means we do not drop any node since the number of nodes in this layer is too small (6 nodes). Normalization is also an important procedure in FCNN, which is normally used in order to avoid the cases where the NN parameters vanish or explode. Batch normalization [24] is very efficient and hence we applied it to all the layers except the output. Equation 5 is the core operation in batch normalization.

$$y = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} * \gamma + \beta \tag{5}$$

x represents a batch, $E(x)$ and $Var(x)$ are the mean and the variance of x respectively, ϵ is added to ensure numerical stability, and β and γ (affine= *True*) are two learnable parameters. Also by default, during training this layer keeps running estimates of its computed mean and variance (track_running_stats= *True*), which are then used for normalization during evaluation. The running estimates are kept with a default momentum of 0.1[2]. After normalization, a linear layer is added (Eq. 6):

$$y = W^{tr}.x + b \tag{6}$$

where W is a learnable parameter matrix, x is the batch, . is the dot product and b is a learnable bias vector. For instance, the first linear layer model connects the input layer (12 nodes) to the first hidden layer (1000 nodes) as shown in Fig. 5. Given a batch size = 64, hence the dimension of matrix W becomes (12, 1000), whereas the dimensions of x are (12, 64) and those of b are (1000, 64).

After adding the linear layer, we introduce a non-linearity by applying an activation function. In our case, it is the ReLU (Rectified Linear Units) function [25]. ReLU is simply $max(0, y)$, to get rid of negative values.

As mentioned before, the model contains two phases: training and validation (see Algorithms 1 and 2). Note that the backward pass is applied during the training phase only; Where the parameters of the model are updated in order to minimize the loss function. The validation phase, however, contains only a forward pass. Note also the Dropout is turned off during the validation.

[1] Regularization is used in order to give the model the ability to generalize on unseen datasets.

[2] Momentum is a hyperparameter, i.e. it can be tweaked. However, the value of 0.1 is generally adopted in the literature [24].

Algorithm 1 The training algorithm

INITIALIZATION

- Select initial values: learning rate ($\alpha = 10^{-5}$), batch size = 64, num_epochs=1.

for epoch in num_epochs **do**

for mini-batch x in training set **do**

FORWARDS PASS

- Add an embedding layer for SNR

- Concatenate the output of the embedding layer with the values of the six detectors (fig. 5)

- Starting from the input layer (fig. 5), apply in sequence: eq. 5, then ReLU, then eq. 6 and Dropout(p) to do a forward pass trough the network.

BACKWARDS PASS

- Calculate the loss function according to eq. 4

- layers $\leftarrow$ [output layer, hidden layer 2, hidden layer 1]

for layer in layers **do**

- Compute the derivatives of the loss function with respect to the weight matrix W and bias vector b connecting it to the previous layer

- Update the weights and biases according to mini-batch gradient descent [26]:

$W \leftarrow W - \alpha * \frac{dloss}{dW}$

$b \leftarrow b - \alpha * \frac{dloss}{db}$

end for

end for

Calculate the average loss value (i.e. training epoch loss)

end for

Table 1 Accuracy as function of different model architectures

Accuracy			
1 layer, 7 nodes	1 layer, 20 nodes	2 layers, [5,5] nodes	2 layers, [1000, 500] nodes
0.83	0.84	0.87	0.96

Algorithm 2 The validation algorithm

for epochs in num_epochs **do**

for mini-batch x in validation set **do**

FORWARDS PASS

- Add an embedding layer for SNR

- Concatenate the output of the embedding layer with the values of the six detectors (fig. 5)

- Starting from the input layer (fig. 5), apply in sequence: eq. 5, then ReLU, then eq. 6 to do a forward pass trough the network.

- Calculate the loss function according to eq. 4

end for

- Calculate the average loss value (i.e. validation epoch loss)

- Calculate the confusion matrix

end for

4 Results

4.1 Model Tweaking

We tested several model architectures with various numbers of layers and different number of nodes per layer.

The results reported in Table 1 are after one epoch of training, since the accuracy was almost independent from the number of epochs. We conducted our experiments on a cloud AWS (Amazon Web Service) machine equipped with a k80 GPU (12 GB integrated RAM; 5.6 TFLOPS [27]). It is clear that increasing the number of layers and the number of nodes per layer leads to better accuracy. However, we did not notice an accuracy improvement with a number of layers more than two. Also, we increased the number of nodes to the maximum value allowed by the machine RAM. In addition to the number of layers and the number of nodes, there are other hyperparameters to tweak. The most important one is the learning rate. We applied the methodology suggested in [28] in order to select a learning rate which minimizes the loss function. The result is illustrated in Fig. 8. We obtained this figure by applying algorithm 1 on a small percentage of the training set (5% in our case).

According to [28], the learning rate should be selected from the decreasing zone in Fig. 8. That is, in the range $[10^{-5}, 10^{-1}]$. In our experiments we used the value 10^{-5}.

Other parameters are: batch size, momentum, epsilon, dropout probability and the length of the embedding vector.

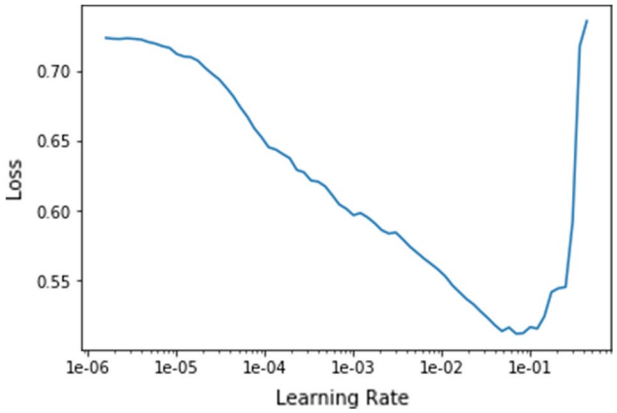
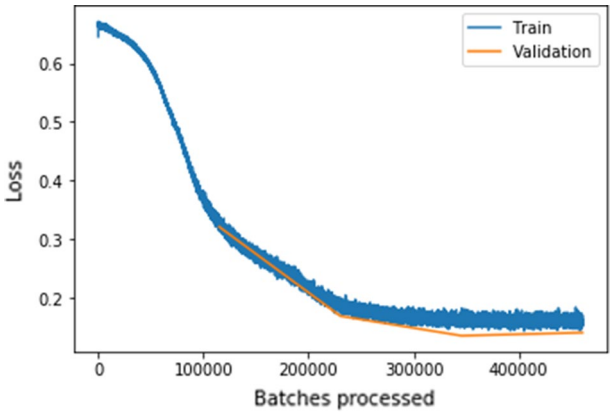


Fig. 8 Selection of the learning rate

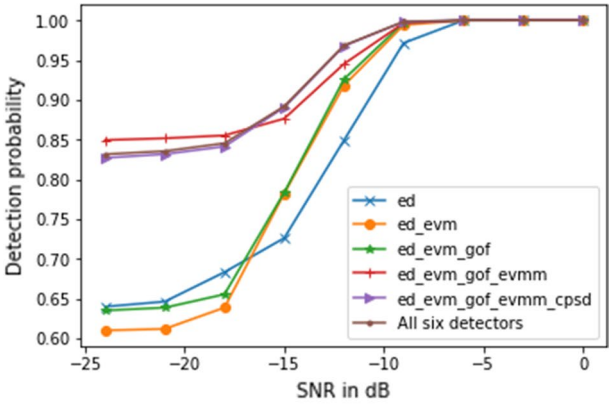
Fig. 9 The loss value as function of the processed batches



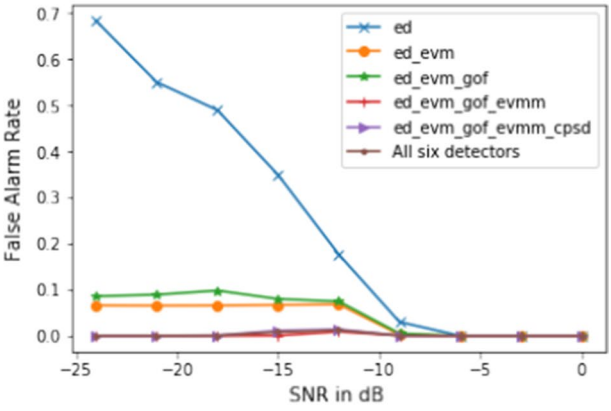
For the batch size, we selected a value of 64 (a larger value can be used but this requires more RAM). For the embedding vector length, the best practice [21] is to reduce the dimension of the categorical input vector (SNR vector in Fig. 5). Hence, any value less than 9 is acceptable. In our experiments, we fixed this value to 6. For the remaining parameters, we used momentum = 0.1 ([26]), epsilon = 10^{-5} (this should be a number close to 0 [24]) and dropout probability $p = 0.001$ for the hidden layer 1 and $p = 0.01$ for the hidden layer 2 (p should be a small percentage of the nodes' layer). With these parameters, we obtained a high accuracy value (0.96) for the model architecture with 2 layers, [1000, 500] nodes. Also, as illustrated in Fig. 9, validation and training losses are very close which means that our model does not over-fit, i.e. it can generalize well to any dataset.

4.2 Sensing Performance Evaluation

In this section, we present results obtained from our model. We emphasize on two performance measures: the probability of detection (PD) and the false alarm rate (FAR). Our dataset contains six detectors which are: ED, ACD, EVM, EVMM, CPSD and GoF. We may present results for any combination among these detectors; However this will be a time



(a) The evolution of PD in terms of SNR



(b) The evolution of FAR in terms of SNR

Fig. 10 Evaluation of PD and FAR in terms of SNRs

consuming. Instead, we take the following set of combinations where ED is common in all the adopted combinations: {*ED*, *ED* − *EVM*, *ED* − *EVM* − *GoF*, *ED* − *EVM* − *GoF*− *EVMM*, *ED* − *EVM* − *GoF* − *EVMM* − *CPSD*,all detectors}. Our assumption comes form the fact that ED is the classical detector in SS and is widely considered as the reference one, thus ED is common in all the considered combinations.

Figure 10 shows the evolution of PD and FAR of ANN-based HSS detector in terms of SNR for all the adopted combinations. Noting that adopting ED solely reflects the classical case when ANN is used to train/validate only one detector, thus it can be considered as the reference of the non-HSS. However, for the combination *ED* − *EVM*, PD increases from 0.6 at SNR = − 24 dB to a value greater than 0.95 at SNR of − 12 dB. This evolution of PD is accompanied with a decrease of FAR from 0.06 at SNR = − 24 dB to a value less than 0.1 at − 12 dB. On the other hand, for the ANN-based ED (no HSS is adopted) PD increases from 0.65 to 0.85 for the SNR range [− 24 ; − 12] dB, while FAR presents very high values compared to *ED* − *EVM* on such SNR range.

Furthermore, Fig. 10 shows that PD increases with the number of used detectors, whereas FAR decreases with the number of used detectors. When three detectors are used, i.e. *ED* − *EVM* − *GoF*, PD achieves 0.92 at − 12 dB and FAR becomes less

than 0.06 for the same SNR. These two performance indicators, PD and FAR, become respectively higher than 0.95 and less than 0.001 when six detectors are used. This fact reflects the efficiency of the hybrid sensing in terms of both protecting PU form the interference (when PD is high) and exploiting the available spectrum resources (when FAR is low).

However, for very low SNR, i.e. − 24 dB, PD is above 0.825 with a FAR less than 0.001, which reveals the high robustness of such a hybrid detector in achieving good performance when the other techniques fail.

In Fig. 11, we present the average values of PD and FAR over all SNRs. The average could be interpreted as the robustness of the proposed technique in terms of SNR. In fact, the data corresponding to H_0 are noise-only related and not impacted by the SNR, thus their detectors scattering remains stable in the space independently of the SNR. On the another hand, the data under H_1 is PU signal dependent, and subsequently it is related to the SNR of the received PU signal. Hence, the performance analysis presenting the average PD and FAR gives us an in-depth view on the efficiency of the proposed technique to distinguish between H_0 and H_1, for wide range of SNR ([− 24 ; 0] dB). For the case where no HSS is used, i.e. only ED is used, thre average PD is around 0.84 for an average FAR of 0.25 as shown in Fig. 11 respectively. In contrast, for HSS when the number of used detectors increases the average PD increases accordingly, whereas the average FAR decreases. An average PD higher than 0.93 is observed when more than 3 detectors are used, while an almost zero FAR is obtained.

5 Conclusion

In this paper, we presented hybrid spectrum sensing (HSS) technique using artificial neural network (ANN). Instead of using one detection method as per the classical spectrum sensing, several test statistics (TSs) of several detectors are combined using ANN. ANN system is trained with the TSs of the used detectors for the noise-only case and for the case where PU is active. The numerical results corroborate the efficiency of the proposed HSS compared to the non hybrid detection technique, where ANN is trained with the TS on only one detector. In addition, the results proved that the detection outcome becomes more reliable as the number of detectors increases.

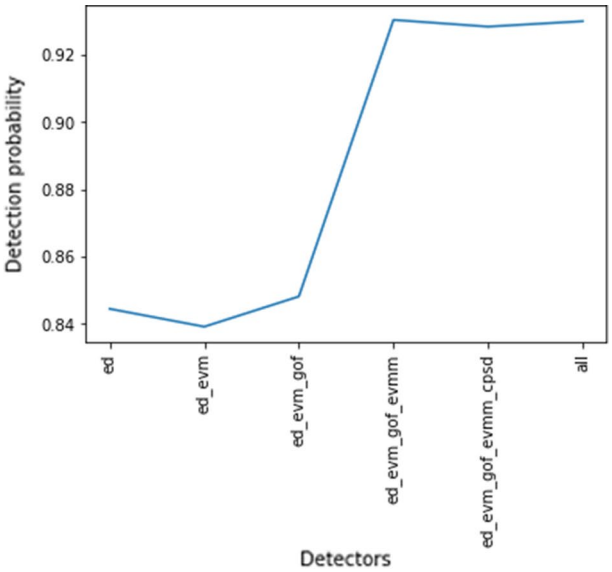
Appendix: Mathematical Formulae of Adopted Detectors

Energy detector (ED) is defined as the sum of the square modulus of the received signal:

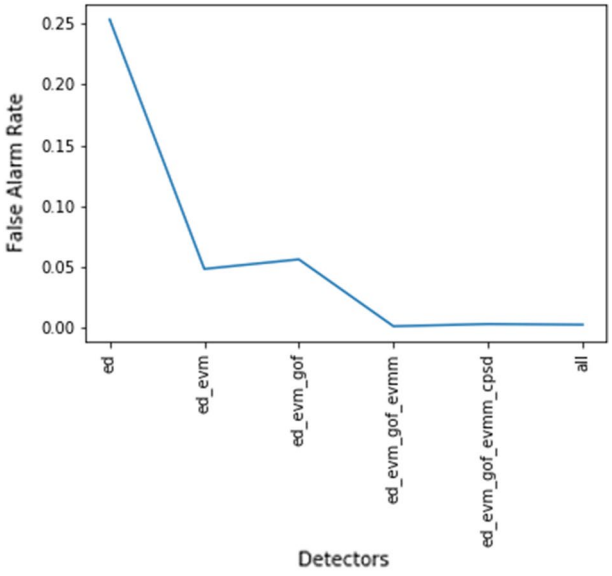
$$T_{ED} = \frac{1}{N} \sum_{1}^{N} |r(n)|^2 \tag{7}$$

where N is the number of received samples.

Autocorrelation detector (ACD) consists of evaluating the inter-sample correlation of the received signal, and is defined as follows:



(a) Average detection probability



(b) Average False Alarm Rate

Fig. 11 The average PD and FAR for the SNR range [− 24 ; 0] dB for the used combination in the proposed ANN-based HSS technique

$$T_{ACD} = \frac{1}{N_s N T_{ED}} \sum_{l=1}^{N_s-1} Re\left\{ \sum_{n=1}^{N} r(n) r^*(n-l) \right\} \tag{8}$$

where $*$ stands for the congugate operation, N_s is the number of samples per symbol and σ_w^2 is the AWGN noise variance.

The CSPD detector evaluates the non-flatness of the noise in frequency domain and is given by [19, eq. 26]:

$$T_{CPSD} = \frac{2}{N^2\sigma_w^2} \sum_{k=1}^{N/2} \left(\frac{N}{2} - k + 1 \right) \frac{|R(m)|^2 + |R(-m+1)|^2}{2} \tag{9}$$

where $R(m)$ is the discrete Fourrier transform of $r(n)$.

EVM consists of finding the maximum eigenvalue of the covariance matrix R_∇ of $\nabla(n)$ which is a set of shifted versions of $r(n)$.

$$T_{EVM} = \lambda_{max} = ||\lambda_1, \lambda_2, \ldots, \lambda_L||_\infty \tag{10}$$

where $\lambda_i,\ i \in [1, L]$ is the ith eigenvalue of R_∇, L is related to the number of shifted versions of $r(n)$, and $||\cdot||_\infty$ is the ∞ norm.

Similarly to EVM, EVMM is evaluated based on the ratio of the maximal eigenvalue to the minimal eigenvalue of $R_\dagger$:

$$T_{EVMM} = \frac{\lambda_{max}}{\lambda_{min}} \tag{11}$$

where $\lambda_{min} = min\{\lambda_1, \lambda_2, \ldots, \lambda_L\}$

Finally, T_{GoF} consists of detecting the presence of PU signal by determining whether the received samples are drawn from the noise distribution with a Cumulative Distribution Function F [29]:

$$T_{GoF} = -\sum_{n}^{N} \left[\frac{\log(F\{r(n)\})}{N - n + 1/2} + \frac{\log(1 - F\{r(n)\})}{n - 1/2} \right] \tag{12}$$

References

1. Mitolal, J. (1999). Cognitive radio: Making software radios more personal. *IEEE Personal Communication*, *6*(4), 13–18.
2. Yucek, T., & Arslan, H. (2009). A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE Communication Surveys & Tutorials*, *11*(1), 116–130. First Quarter.
3. Clancy, C., Hecker, J., Stuntebeck, E., & O'Shea, T. (2007). Applications of machine learning to cognitive radio networks. *IEEE Wireless Communications*, *14*(4), 47–52.
4. Thilina, K. M., Choi, K. W., Saquib, N., & Hossain, E. (2013). Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, *31*(11), 2209–2221.
5. Lu, Y., Zhu, P., Wang, D., & Fattouche M. (2016). Machine learning techniques with probability vector for cooperative spectrum sensing in cognitive radio networks. In *2016 IEEE wireless communications and networking conference* (pp. 1–6).
6. Vyas, M. R., Patel, D. K., & Lopez-Benitez, M. (2017). Artificial neural network based hybrid spectrum sensing scheme for cognitive radio. In *2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)* (pp. 1–7).
7. Tang, Y., Zhang, Q., & Lin, W. (2010). Artificial neural network based spectrum sensing method for cognitive radio. In *2010 6th international conference on wireless communications networking and mobile computing (WiCOM)* (pp. 1–4).

8. Li, Z., Wu, W., Liu, X., & Qi, P. (2018). Improved cooperative spectrum sensing model based on machine learning for cognitive radio networks. *IET Communications*, *12*(19), 2485–2492.
9. Guo, C., Jin, M., Guo, Q., & Li, Y. (2018). Spectrum sensing based on combined eigenvalue and eigenvector through blind learning. *IEEE Communications Letters*, *22*(8), 1636–1639.
10. Shah, I., & Koo, H. A. (2018). Reliable machine learning based spectrum sensing in cognitive radio networks. *Wireless Communications and Mobile Computing, 2018*
11. Ahmad, H. B. (2019). Ensemble classifier based spectrum sensing in cognitive radio networks. *Wireless Communications and Mobile Computing, 2018*
12. Molina-Tenorio, Y., Prieto-Guerrero, A., Aguilar-Gonzalez, R., & Ruiz-Boqué, S. (2019). Machine learning techniques applied to multiband spectrum sensing in cognitive radios. *Sensors, 19*(21).
13. Shirazi, S. F., Shirazi, S. H., Shah, S. M., & Shahid, M. K. (2012). Article: Hybrid spectrum sensing algorithm for cognitive radio network. *International Journal of Computer Applications*, *45*(17), 25–30.
14. Moghimi, F., Schober, R., & Mallik, R. K. (2010). Hybrid coherent/energy detection for cognitive radio networks. In *2010 IEEE global telecommunications conference GLOBECOM 2010* (pp. 1–6).
15. Cardenas-Juarez, M., Ghogho, M., Pineda-Rico, U., & Stevens-Navarro, E. (2016). Improved semi-blind spectrum sensing for cognitive radio with locally optimum detection. *IET Signal Processing*, *10*(7), 524–531.
16. Digham, F., Alouini, M.-S., & Simon, K. (2007). On the energy detection of unknown signals over fading channels. *IEEE Transactions on Communications*, *55*(1), 21–24.
17. Naraghi-Poor, M., & Ikuma, T. (2010). Autocorrelation-based spectrum sensing for cognitive radio. *IEEE Transactions on Vehicular Technology*, *59*(2), 718–733.
18. Zeng, Y., & Liang, Y.-C. (2009). Eigenvalue-based spectrum sensing algorithms for cognitive radio. *IEEE Transactions on Communications*, *57*(6), 1784–1793.
19. Nasser, A., Mansour, A., Yao, K. C., Abdallah, H., & Charara, H. (2017). Spectrum sensing based on cumulative power spectral density. *EURASIP Journal on Advances in Signal Processing*, *2017*(1), 38.
20. Teguig, D., Le Nir, V., & Scheers, B. (2015). Spectrum sensing method based on the likelihood ratio goodness of fit test. *IEEE Electronic Letters*, *51*(3), 253–255.
21. Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv, http://arxiv.org/abs/1604.06737.
22. Zhu, D., Yao, H., Jiang, B., & Yu, P. (2018). Negative log likelihood ratio loss for deep neural network classification. arXiv, http://arxiv.org/abs/1804.10690.
23. Labach, A., Salehinejad, H. & Valaee, S. (2019). Survey of dropout methods for deep neural networks. arXiv, http://arxiv.org/abs/1904.13310.
24. Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv, http://arxiv.org/abs/1502.03167.
25. Arora, R., Basu, A., Mianjy, P. & Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. arXiv, http://arxiv.org/abs/1611.01491.
26. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv, http://arxiv.org/abs/1609.04747.
27. Markidis, S., Der Chien, S. W., Laure, E., Peng, I. B., & Vetter, J. S. (2018). Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)* (pp. 522—531).
28. Smith, L. N. (2015). Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)* (pp. 464–472).
29. Zhang, G., Wang, X., Liang, Y.-C., & Liu, J. (2010). Fast and robust spectrum sensing via Kolmogorov–Smirnov test. *IEEE Transactions on Communications*, *58*(12), 3410–3416.

A. Nasser received the B.Sc. in Electronics and the M.Sc. in Signal, Telecom, Image and Speech, of the Lebanese University in 2010 and 2012 respectively. From 2012 to 2013, he was assistant at the Department of Computer Science, American University of Culture and Education (AUCE), Lebanon. From April 2013 to December 2016, he has been a PhD student at the Université de Bretagne Occidendale (UBO), France. From June 2017 to May 2019, he has been post-doc researcher at Ensta-Bretagne, France. Since June 2019, he has been a research fellow at Ensta-Bretagne. He has been a visiting professor to Polytech-Nantes in October 2020. He also holds the post of assistant professor at AUCE, Lebanon since January 2017. Abbass NASSER is participating in the supervision of several PhD projects in Electronics and Digital Communication. He has more than 35 research papers published in International Journals and Conferences. His current research interests include cognitive radio, signal detection and estimation, Internet of Things and wireless sensor networks.

M. Chaitou received an M.S. degree in networking from the University of Pierre et Marie Curie, France, in 2003. In 2006, He obtained a Ph.D. degree in computer science from Telecom SudParis (TSP), France. From 2007 to 2010, He worked on the development of IP backbone networks for Orange Labs and Bouygues Telecom respectively. Since 2010, He occupies the position of assistant professor at the Lebanese university, faculty of science. His actual research interests include: applications of data science in telecommunications, cloud computing and network security.

A. Mansour received the M.S. in electronic electric engineering from Lebanese University in September 1992, the M.Sc. and Ph.D. degrees in signal, image and speech processing from INPG, Grenoble-France, in July 1993 and January 1997, respectively, and the HDR degree (Habilitation à Diriger des Recherches, this is the highest of the higher degrees in the French system) from UBO, Brest-France, in November 2006. He held many positions such as: Postdoctoral at LTIRF-INPG (Grenoble-France), Researcher at BMC – RIKEN (Nagoya-Japan), Teacher-Researcher position at ENSIETA (Brest-France), Senior Lecturer at ECE Curtin University (Perth-Australia), Invited Professor at ULCO (Calais-France), Professor at Tabuk University (Tabuk-KSA). Actually, he holds a post of Professor at ENSTA-Bretagne (Brest-France). He published numerous refereed publications, he is the author and co-author of several books or book chapters. During his career, he had successfully supervised several research associates and PhD and MSc students. He is a senior member of IEEE and was the vice president for IEEE signal processing society in Western Australia for two years. He had also been the lead guest editor for the EURASIP Journal on Advances in Signal Processing. He is interested in Blind Source Separation, High Order Statistics, signal processing, robotics, telecommunication, biomedical engineering, electronic warfare and cognitive radio.

K. C. Yao received the Ph.D. degree in optical signal processing and computer sciences from University Louis Pasteur, Strasbourg, in 1990. After his postdoctoral research period on optical neural networks with Ecole Nationale Supérieure des Téelécommunications (ENST), IMT Atlantique, Brest, He joined the French Naval Academy as an Assistant Professor in statistical signal processing in 1992. Since 2001, he has been an Assistant Professor with the Université de Bretagne Occidentale (University of Western Brittany), Brest, France. His research interest focuses on Pattern recognition in sonar imagery and blind signal separation in underwater acoustics channel. He is member of the laboratory Lab-STICC CNRS UMR 6285, Team Security, Intelligence and Integrity of Information (SI3). His present research interests concern wireless communications, blind interception of digital communication signals, and cognitive radio.

H. Charara has received an M.S. degree in Computer and Communications Engineering from the Lebanese University, Lebanon in 2002, and an M.S. degree in Networking and Telecommunications from the Institut National Polytechnique (INP-ENSEEIHT), France, in 2003. In 2007, He obtained a Ph.D. degree in Network, Telecom, Systems and Architecture from INP - IRIT, France. From 2006 to 2009 He worked for AIRBUS and THALES AV avionics as a R&D Engineer and PM. He contributed to the implementation & development of the Real time embedded AFDX networks such as for A380, A400M and Soukhoï RRJ programs. From 2009 to 2010, He worked for the ARECS GmbH - München where He was involved in the GPS/Galileo receivers modeling and simulations. Since 2010, He is an assistant professor at the Lebanese University. He is working in several research fields including: Spectrum Sensing, Traffic engineering of real time network, Avionics and wireless sensor networks, Neural networks.