Long Short-Term Memory based Spectrum Sensing Scheme for Cognitive Radio

Nikhil Balwani*, Dhaval K. Patel*, Brijesh Soni*, Miguel López-Benítez†‡

*School of Engineering and Applied Science, Ahmedabad University, India
†Department of Electrical Engineering and Electronics, University of Liverpool, United Kingdom
‡ARIES Research Centre, Antonio de Nebrija University, Spain
Email: *{nikhil.b.btechi16, dhaval.patel, brijesh.soni}@ahduni.edu.in, †m.lopez-benitez@liverpool.ac.uk

***Abstract*—The application of machine learning models to spectrum sensing in cognitive radio is not uncommon in literature, but most of these models fail to consider temporal dependencies in the signal. In this paper, the temporal correlation among the spectrum data is exploited using a Long Short-Term Memory (LSTM) network. More specifically, the previous sensing event is fed along with the present sensing event to the LSTM model. The proposed sensing scheme is validated based on empirical data of various radio technologies. The proposed LSTM model is compared with other machine learning algorithms in terms of classification accuracy. Furthermore, the proposed scheme is also compared with other spectrum sensing techniques. Results indicate that the proposed scheme improves the detection performance and classification accuracy at low signal-to-noise ratio regimes. Moreover, it is observed that the achieved improvement is obtained at the expense of longer training time and nominal increase in execution time.**

***Index Terms*—Cognitive Radio, Spectrum sensing, ANN, RNN, LSTM, Machine Learning.**

I. Introduction

Due to the proliferation of mobile devices, the demand for spectrum has escalated. Dynamic Spectrum Access (DSA) / Cognitive Radio (CR) has emerged as a promising solution to bridge the gap between spectrum availability and its demand growth. It aims at reusing temporarily unoccupied frequency bands, known as *spectrum holes* or *white spaces*, in an opportunistic manner, ensuring that the licensed user does not face any harmful interference [1]. For an interference-free access, the secondary user should be able to identify vacant frequency bands in the spectrum, use those bands upon availability, and vacate them as soon as a primary user requests access. This calls for high reliability and efficient utilization at the same time [1]. Various sensing algorithms have been proposed in the literature to identify the presence or absence of primary users in a given band.

Spectrum sensing algorithms can be categorized as parametric or non-parametric. Parametric sensing schemes work based on some prior information about the primary user activity. However, in most real-world applications, no prior information is available about the primary user, and thus non-parametric sensing schemes are preferred [2]. Energy detection based approaches are common due to their low computational complexity and ease of implementation with empirical setups. However, their performance depends on two key assumptions, the stationarity of noise and the knowledge of its variance [3]. Imperfect knowledge of noise variance leads to a phenomenon called Signal-to-Noise Ratio (SNR) wall [4]. However, various Goodness of Fit (GoF) tests like Anderson Darling test [5], Kolmogorov-Smirnov test [6], Likelihood Ratio Statistics [7] based sensing and machine learning based spectrum sensing schemes are proposed in literature.

In recent past, machine learning algorithms have attracted wide attention from industry and academia in the context of wireless research. The survey in [8] comprehensively describes the CR architectures, reasoning and learning engines and their applications to CR networks using machine learning. Artificial Neural Network (ANN) based spectrum sensing for cognitive radio was carried out in [9]. However in [10], an ANN-based sensing scheme was proposed which used the Zhang statistic [11] and energy values over the sensing events as training features. A detailed study of various machine learning algorithms to identify the spectrum occupancy was provided in [12]. Furthermore, [13] has described various machine learning paradigms for next-generation wireless networks.

As per [14]-[15], traditional ANNs have a shortcoming in a way that they cannot store information due to the absence of memory elements. Furthermore, ANNs are not suited for temporal modelling and for data with long-term dependencies. Moreover, time series data sets can be modelled using Recurrent Neural Networks (RNNs). A major shortcoming of RNNs is that they cannot model long-term dependencies due to the issue of vanishing gradients [16]. In order to overcome the aforementioned shortcomings and to exploit the long-term dependencies in data, LSTM networks are used [17].

To the best of the authors' knowledge, there are very few works in the literature that have used LSTM networks to model the temporal dependencies of spectrum data. In [18], authors proposed a spectrum prediction algorithm using LSTM network whereas, authors in [19] addressed the modulation classification problem using LSTM network. In [20], the authors have used the Taguchi method for hyperparameter optimization of the LSTM network for spectrum prediction. However, the above studies have addressed the spectrum prediction problem and have shown the comparison in terms of accuracy with other machine learning models. By contrast, this work uses LSTM network to address the spectrum sensing problem in CR networks.

The main contributions of this paper are summarized below:

- A novel LSTM based framework for spectrum sensing in CR is proposed in which the previous sensing event is fed along with the present sensing event to exploit the temporal dependencies in the data set. Results indicate remarkable performance improvements even at low SNR.

- In order to make the LSTM model robust and unbiased towards high SNRs, the training data set includes data at very low SNRs. This ensures that the detection performance does not deteriorate at low SNRs.
- The proposed sensing scheme is experimentally validated with spectrum data of various radio technologies captured using an empirical testbed measurement setup. Furthermore, it is compared with other machine learning algorithms in terms of classification accuracy.

The rest of this paper is organized as follows. Section II describes the system model and the preliminaries of LSTM. The proposed RNN-LSTM model for spectrum sensing is discussed in Section III. Section IV comprehensively describes the measurement setup. Results are discussed in Section V. Finally, Section VI draws the conclusions from this work.

II. System Model and LSTM Preliminaries

The problem of spectrum sensing can be formulated as a binary classification problem[1]:

$$\begin{aligned}\mathcal{H}_0 &: y^{<t>} = w^{<t>} \\ \mathcal{H}_1 &: y^{<t>} = h^{<t>}x^{<t>} + w^{<t>}\end{aligned} \tag{1}$$

where $x^{<t>}$ denotes the primary user signal, $w^{<t>}$ is white Gaussian noise with zero mean and variance σ^2 and $y^{<t>}$ is the received signal at t^{th} time instant. $\mathcal{H}_0$, the null hypothesis indicates the noise samples while $\mathcal{H}_1$, the alternate hypothesis indicates the presence of primary user signal along with noise at t^{th} instant. In order to exploit the temporal dependencies, the previous sensing event is fed along with the current sensing event and thus the received signal, in general, can be expressed as:

$$y = \big[\underbrace{y^{<1>}y^{<2>}\dots y^{<N>}}_{\text{Previous sensing event}}\underbrace{y^{<N+1>}y^{<N+2>}\dots y^{<2N>}}_{\text{Current sensing event}}\big]^T$$

where N is the signal sample size and $[\xi]^T$ denotes the transpose of vector ξ. Fig 1. shows the internal structure of an LSTM cell [17], where $y^{<t>}$ is the input, $a^{<t>}$ is the output of the LSTM cell, $a^{<t-1>}$ is the previous LSTM output, and $c^{<t>}$ and $c^{<t-1>}$ are the current and previous cell states, respectively. σ_u, σ_f, and σ_o are the values of the update, forget and output gates, respectively, $\tanh$ is the activation function, $\odot$ is the Hadamard product and $\oplus$ denotes element-wise addition.

A. Key Elements

An LSTM cell has these key elements:

1) Update gate: Decides when to update the current cell state, denoted as the output of σ_u.
2) Forget gate: Decides when to discard the current cell, denoted as the output of σ_f.
3) Output gate: Controls the output, denoted as the output of σ_o.

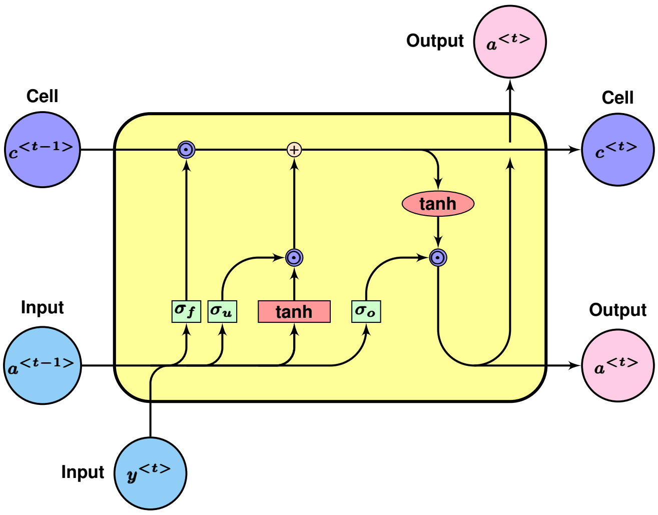


Figure 1: Internal structure of LSTM cell with one hidden unit.

B. Functioning

Using the $\tanh$ activation function:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2}$$

a vector of candidate values, $\tilde{c}^{<t>}$, is created in order to update the cell state:

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{3}$$

where b_c is the bias term. The values for the update, forget and output gates are calculated by applying a sigmoid activation:

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{4}$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{5}$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{6}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

where W_u, W_f, and W_o are the weight matrices and b_u, b_f, and b_o are the bias terms. An elementwise product is taken between the forget gate (Γ_f) and the previous cell state $c^{<t-1>}$, and between the update gate (Γ_u) and the candidate vector for updation $\tilde{c}^{<t>}$. Output $a^{<t>}$ is the element-wise product between the output gate (Γ_o) and the hyperbolic tangent of candidate vector $c^{<t>}$:

$$c^{<t>} = \Gamma_u \odot \tilde{c}^{<t>} + \Gamma_f \odot c^{<t-1>} \tag{8}$$

$$a^{<t>} = \Gamma_o \odot \tanh(c^{<t>}) \tag{9}$$

III. Proposed RNN-LSTM Scheme

Traditional ANNs described in [9] have no memory elements and hence lack the ability to store data. Therefore, it is necessary to modify the structure of neural networks to have feedbacks between successive timestamps [14], [15]. Fig. 2 shows the proposed LSTM model comprising of LSTM cells (as described in Section II) and an output cell which goes through the sigmoid activation.

[1] Notations in system model are modified in order to have consistency with LSTM notations.

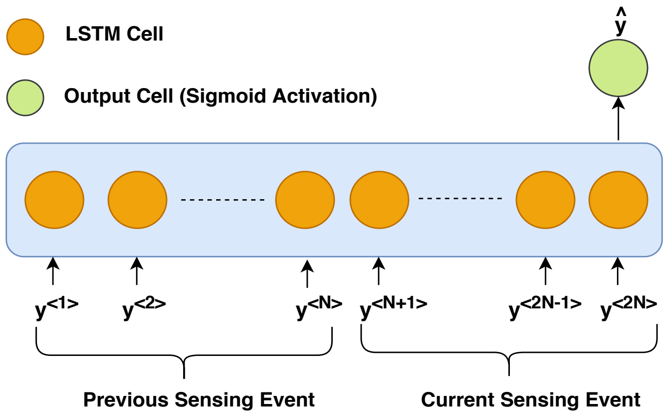


Figure 2: Proposed sensing model.

Algorithm 1 Dataset construction

```
procedure CREATE DATASET(Data, N, Label)
    size ← $\frac{\text{length(Data)}}{\text{N}}$
    PU_dataset ← zero matrix of dimensions size × N
    for $SNR$ ← -20 to 4 dB do
        noisy_signal ← Data + AWGN                    ▷ SNR is achieved
        for $i$ ← 1 to size do
            signal ← $(i)^{\text{th}} N$ samples from noisy_signal
            PU_signal[i] ← signal                      ▷ Row-wise assignment
    return PU_signal                 ▷ The primary user signal is returned
```

A. Dataset Construction

In this study, the proposed LSTM model is trained and validated based on spectrum data. The data is captured through an empirical setup, a detailed discussion of which is provided in Section IV. From the captured data, the clean primary user signal is acquired and its power σ_x^2 is measured. In order to achieve a given SNR γ, the required power of noise to be added is calculated using the relation $\sigma_w^2 = \sigma_x^2/\gamma$ [7]. Additive White Gaussian Noise (AWGN) sequence of the power level is generated and added to the signal.

For a sample of size N, the signal will thus be a vector with $2N$ timestamps.

$$y = \left[y^{<1>} y^{<2>} \dots y^{<2N>}\right]^T$$

Each signal vector with sample size N is considered as a sensing event and hence is taken as a training example for the LSTM model. For this study, 200,018 examples are maintained for AWGN and 200,018 examples are maintained for primary user signal in the SNR range -20 dB to 4 dB. Thus, the dataset contains an equal number of primary user signal and AWGN sequence examples.

In order to make the LSTM scheme robust and unbiased towards high SNRs, the training data set includes data at very low SNRs. Algorithm 1 and Fig. 3 show the data construction process. The generated data are divided into three classes, training (60%), validation (20%) and test (20%) datasets.

B. LSTM Training and Model Selection

Keras library with TensorFlow backend is used to create and train models. As shown in Fig. 4, the training dataset examples, which comprise of 60% of the total examples, are fed in batches to different LSTM models, the error is backpropagated during the training procedure, the gradients are calculated and the parameters are updated, as illustrated in Algorithm 2. Accuracies of these models on the training and validation sets are evaluated. The training set performance of a given model does not always generalize to other datasets as big models tend to overfit the training data. Thus, validation set accuracies are considered for choosing the best model.

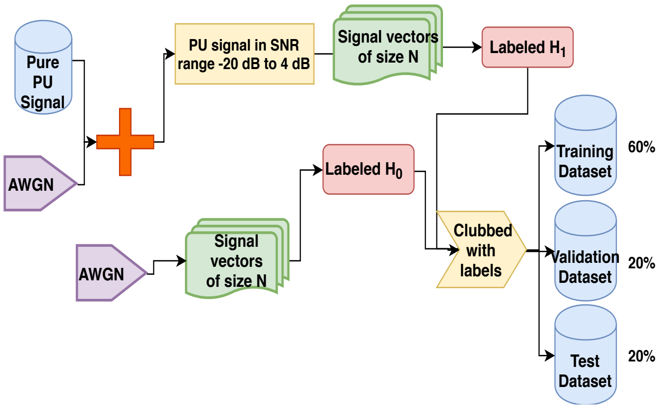


Figure 3: Dataset construction.

Algorithm 2 LSTM training

```
procedure TRAIN(Epochs, Batch_size, X, y, $\alpha$)
    for $i$ ← 1 to Epochs do
        s_event, label ← extract(Dataset, Batch_size)
  ▷ Random training examples are extracted according to the batch size
        Output ← Forward Propagate(LSTM_model, s_event)
        Error ← Backward Propagate(LSTM_model, label, output)
        Parameters ← Update(error, LSTM_model,$\alpha$)    ▷ Parameters are
updated according to the learning rate $\alpha$
```

Table I: Training/validation set accuracies of different models.

Number of hidden units	Epochs	Training accuracy	Validation accuracy
1	5	88.89%	88.11%
	10	88.39%	88.75%
	15	88.66%	88.50%
128	5	92.59%	87.80%
	10	95.63%	86.03%
	15	97.06%	85.20%
256	5	95.60%	86.81%
	10	98.56%	84.81%
	15	99.31%	84.66%

As the number of hidden units is increased, training accuracy goes up but validation accuracy declines. This happens when the model overfits the training dataset. To avoid overfitting, we have evaluated training and validation accuracies for models with different numbers of hidden units (see Table I). It can be seen that the validation accuracy is maximum for the LSTM network with one hidden unit.

C. Evaluation of Performance Metrics

This is the final phase where the proposed sensing scheme is evaluated. Algorithm 3 shows the evaluation procedure to calculate probability of detection P_d and probability of false alarm P_f. Signal data from the test dataset are fed one by

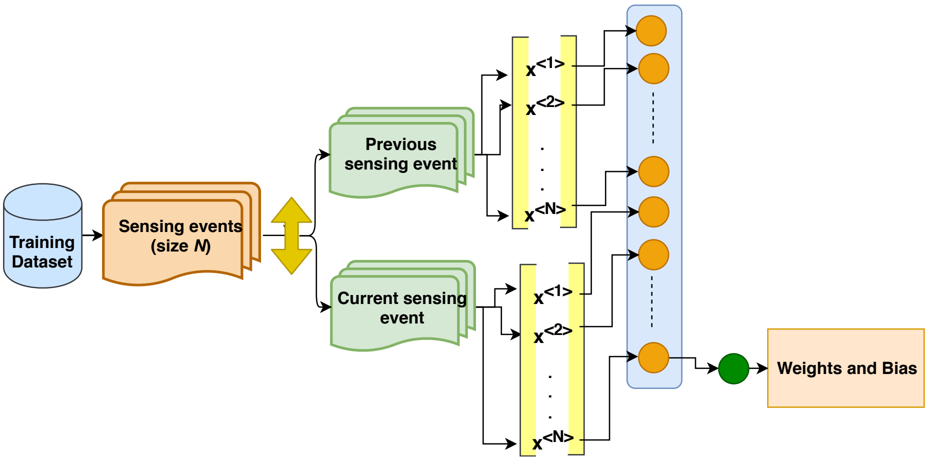


Figure 4: LSTM training model considered in this work.

Algorithm 3 Evaluation of the model

```
procedure EVALUATE(LSTM_model, Dataset)
    for i ← 1 to length(PU_signal) do
        s_event, label ← extract(Dataset, 1)
                                    ▷ Test examples are extracted one by one
        $\mathcal{H}_0$_examples ← 0
        $\mathcal{H}_0$_misclassified ← 0
        $\mathcal{H}_1$_examples ← 0
        $\mathcal{H}_1$_correct ← 0
        Output ← Forward_Propagate(LSTM_model, s_event)
        if Label is $\mathcal{H}_1$ then
            $\mathcal{H}_1$_examples ← $\mathcal{H}_1$_examples + 1
            if Output is $\mathcal{H}_1$ then
                $\mathcal{H}_1$_correct ← $\mathcal{H}_1$_correct + 1
        if Label is $\mathcal{H}_0$ then
            $\mathcal{H}_0$_examples ← $\mathcal{H}_0$_examples + 1
            if Output is $\mathcal{H}_1$ then
                $\mathcal{H}_0$_misclassified ← $\mathcal{H}_0$_misclassified + 1
        $P_d \leftarrow \frac{\mathcal{H}_1\_correct}{\mathcal{H}_1\_examples}$
        $P_f \leftarrow \frac{\mathcal{H}_0\_misclassified}{\mathcal{H}_0\_examples}$
```

one to the LSTM network and P_d, P_f are calculated. First, the primary user signal vectors at each SNR are forwarded to the LSTM network. The number of times it correctly classifies the signal, i.e. ($\mathcal{H}_1$), divided by the total number of primary user signal examples fed to the network determines P_d. Similarly, AWGN sequence examples are forwarded to the LSTM network and P_f is calculated as the number of times it does not predict $\mathcal{H}_0$ divided by the total number of AWGN sequence examples.

IV. Empirical Measurement Setup

An empirical testbed was deployed for spectrum data acquisition on the roof-top of the School of Engineering and Applied Science, Ahmedabad University. The measurement setup is as shown in Fig. 5. The hardware consists of a digital spectrum analyzer Rigol DSA-875, a Universal Software Radio Peripheral (USRP-N210) with a WBX daughterboard surmounted, two Diamond D-3000N discone antennae and a computer system to interface the hardware and software. The software part includes MATLAB and GNU Radio. Table II shows the tuning parameters of the spectrum analyzer while Table III shows the measured channels and the USRP configuration.

Table II: Tuning parameters of spectrum analyzer.

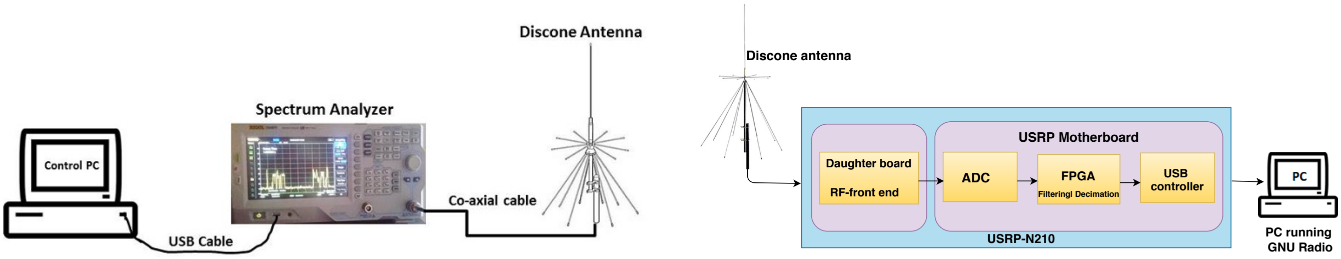
Parameter	Value
Frequency range	75-2000 MHz
Frequency span	45-600 MHz
Frequency bin	Depends on band selected
Resolution Bandwidth-RBW	10 kHz
Video Bandwidth-VBW	10 kHz
Measurement period	5-15 mins
Sweep time	1 second
Scale	10 dB/division
Input attenuation	0 dB
Detection type	RMS detector

Rigol DSA-875 spectrum analyzer supports 601 frequency points. A resolution bandwidth (RBW) of 10 kHz was selected and a sweep period of 1 second was kept. The frequency bins selected in the spectrum analyzer were kept slightly wider than those selected in USRP. With the help of analyzer, channels with high SNR were identified for various radio technologies (see Table III), which were afterwards used to capture primary signal data using the USRP. The acquired data using GNURadio are further processed offline in MATLAB and then the validation of the proposed LSTM based sensing scheme is carried out.

V. Experimental Results

A. LSTM Training Analysis

1) Manipulation in Training data and Training Strategies: The training data cluster was divided into two classes: low SNR class and high SNR class. -20 dB to -6 dB were clubbed and categorized in the low SNR class; similarly, -4 dB to 4 dB were categorized in the high SNR class. In order to perform the training analysis, the proportion of training examples in each of the two classes was varied and consequent variations in P_d values at different SNRs and P_f values were observed. Different compositions of the training dataset were created by varying the ratio of the number of examples in low SNR class to the number of examples in high SNR class. Algorithm 1 is used to construct the datasets and Algorithm 2 is used to



(a) Spectrum analysis

(b) Spectrum data acquisition

Figure 5: Empirical measurement setup used in this work.

Table III: USRP configuration and channels measured in this work.

Radio Technology	Channel Number	f_{start} (MHz)	f_{center} (MHz)	f_{stop} (MHz)	Signal bandwidth (MHz)	Gain (dB)	Decimation Rate	Sampled Bandwidth (MHz)
FM broadcasting	–	96.500	96.700	96.900	0.2	45	64	1
UHF television (Band IV)	U-33	566	570	574	8	45	8	8
E-GSM 900 DL	77	950.2	950.4	950.6	0.2	45	64	1
DCS 1800 DL	690	1839.6	1840.8	1841	0.2	45	64	1

Table IV: False alarm rates for different compositions of the training data cluster (UHF Television, $N = 100$).

Model name	% of examples in low SNR range	% of examples in high SNR range	P_f
10-90	10%	90%	0.0100
20-80	20%	80%	0.0252
30-70	30%	70%	0.0432
40-60	40%	60%	0.0527
50-50	50%	50%	0.0528
60-40	60%	40%	0.0698
70-30	70%	30%	0.0718
80-20	80%	20%	0.0871
90-10	90%	10%	0.1051

train the LSTM network on these datasets. The compositions are evaluated using Algorithm 3 and P_d, P_f are determined. It is evident from Table IV and Fig. 6 that the composition of training set has a significant impact on P_d and P_f. As the percentage of examples in low SNR range is increased, P_f and P_d also increase. At low SNRs, the magnitudes of primary user signal are similar to that of noise. The LSTM network, therefore, finds it difficult to differentiate between primary signal and noise. On the contrary, if the LSTM network is trained on a dataset with a high proportion of low SNR examples, then it is more likely to predict that the primary user is present, which results in higher values of P_f and P_d.

2) Performance Analysis: The proposed scheme was validated on four radio technologies, namely, FM Broadcasting, E-GSM 900 DL, DCS 1800 DL and UHF Television. Fig. 7 shows P_d versus SNR for $N = 100$. To enable a fair comparison with the results of [10], the P_f close to three decimal places was chosen from Table IV. The results show that although the performance of the proposed scheme at high SNRs is almost the same, it significantly outperforms the formerly proposed ANN-based sensing scheme at low SNRs.

3) Training and Execution Time: The ANN based Hybrid Sensing Scheme [10] was trained on 50 epochs. The Naive Bayes classifier was trained with variance smoothing of 10^{-9}. For the Random Forest classifier, the minimum number of samples required to split an internal node was two, and the tree was split until either the leaves had one sample each or all the samples in the leaves were pure. It can be observed from table V that the Naive Bayes algorithm has the lowest training and execution time. Fig. 8 shows the classification accuracies of these machine learning models.

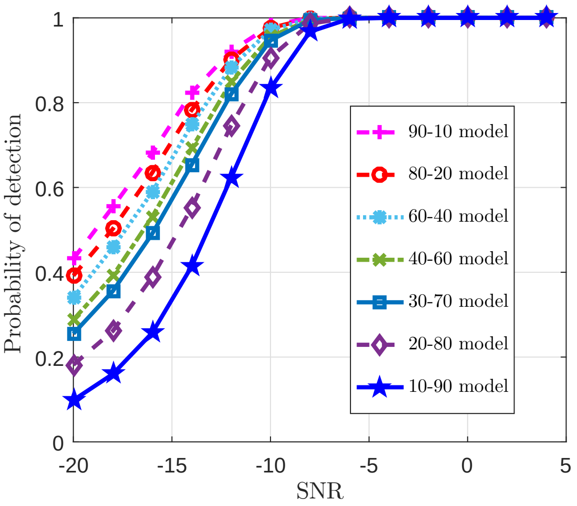
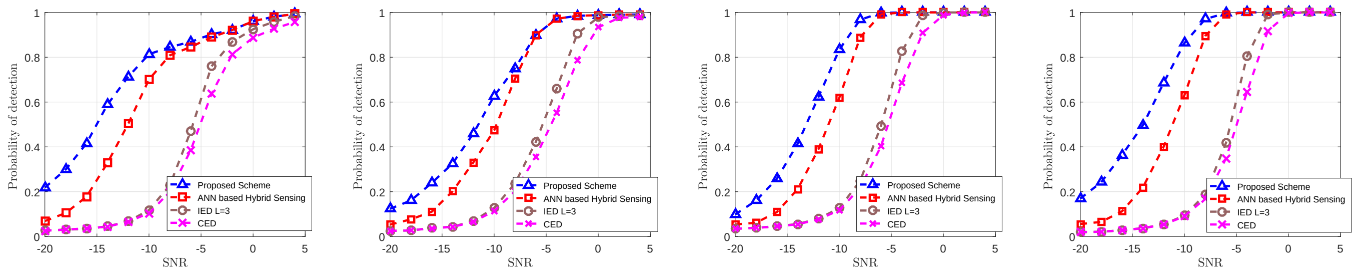


Figure 6: Comparison of different training models (UHF Television, $N = 100$).

Table V: Training and execution times of various algorithms.

Algorithm	Training time (s)	Execution time (ms)
Proposed Algorithm (15 epochs)	345.82	0.788
ANN based Hybrid Sensing (50 epochs)	101.48	0.624
Gaussian Naive Bayes	25.89	0.0391
Random Forest	93.44	0.7158



(a) FM Broadcasting P_f=0.016, N=100

(b) GSM-900 DL P_f=0.015, N=100

(c) DCS-1800 DL P_f=0.022, N=100

(d) UHF Television P_f=0.010, N=100

Figure 7: Detection performance of the considered spectrum sensing methods: proposed LSTM-based scheme, ANN-based scheme from [10], Improved Energy Dection (IED) from [21], and Classical Energy Detection (CED).

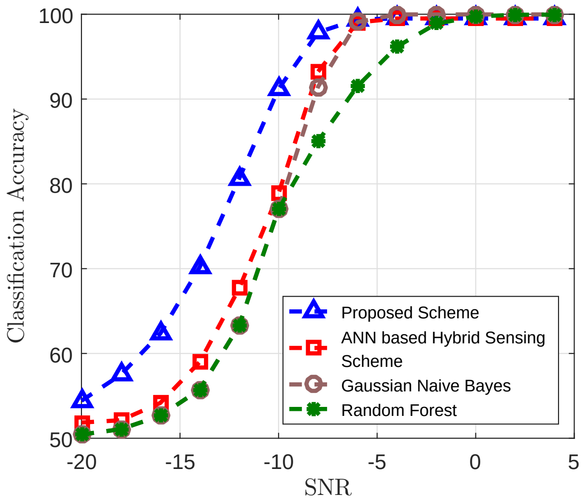


Figure 8: Comparison of classification accuracies of different machine learning models ($N = 100$).

VI. Conclusion

LSTM networks have an excellent ability to learn both long and short term dependencies in the input. In this work, the previous sensing event is fed along with the present sensing event to an LSTM model. The proposed sensing scheme is validated on empirical data of different wireless technologies. Results indicate that the proposed scheme has an improved detection performance and classification accuracy as compared to ANN based hybrid sensing scheme, IED and CED, even under low SNR regime at the expense of longer training time and nominal increase in execution time. The ways to reduce the time consumption of RNN-LSTM networks is an important issue of prospect research.

Acknowledgment

The authors would like to thank financial support received from UKIERI under the DST Thematic Partnership 2016-17 (ref. DST/INT/UK/P-150/2016). The authors also thank Ahmedabad University and University of Liverpool for infrastructure support.

References

[1] S. Haykin, “Cognitive radio: brain-empowered wireless communications,” *IEEE J. Sel. Areas Com.*, vol. 23, no. 2, pp. 201–220, Feb 2005.

[2] S. Haykin, D. J. Thomson, and J. H. Reed, “Spectrum sensing for cognitive radio,” *Proc. IEEE*, vol. 97, no. 5, pp. 849–877, May 2009.

[3] T. Yucek and H. Arslan, “A survey of spectrum sensing algorithms for cognitive radio applications,” *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 116–130, First Quarter 2009.

[4] R. Tandra and A. Sahai, “SNR walls for signal detection,” *IEEE J. Sel. Topics in Sig. Process.*, vol. 2, no. 1, pp. 4–17, Feb 2008.

[5] H. Wang, E. H. Yang, Z. Zhao, and W. Zhang, “Spectrum sensing in cognitive radio using goodness of fit testing,” *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 5427–5430, Nov. 2009.

[6] G. Zhang, X. Wang, Y.-C. Liang, and J. Liu, “Fast and robust spectrum sensing via Kolmogorov-Smirnov test,” *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3410–3416, 2010.

[7] D. K. Patel and Y. N. Trivedi, “LRS-G^2 based non-parametric spectrum sensing for cognitive radio,” in *Proc. CROWNCOM*, 2016, pp. 330–341.

[8] C. Clancy, J. Hecker, E. Stuntebeck, and T. O’Shea, “Applications of machine learning to cognitive radio networks,” *IEEE Wirel. Commun.*, vol. 14, no. 4, pp. 47–52, August 2007.

[9] Y.-J. Tang, Q.-Y. Zhang, and W. Lin, “Artificial neural network based spectrum sensing method for cognitive radio,” in *Proc. WiCOM*. IEEE, 2010, pp. 1–4.

[10] M. R. Vyas, D. K. Patel, and M. López-Benítez, “Artificial neural network based hybrid spectrum sensing scheme for cognitive radio,” in *Proc. PIMRC*, Oct 2017, pp. 1–7.

[11] J. Zhang and Y. Wu, “Likelihood-ratio tests for normality,” *Comput. Stat. Data Anal.*, vol. 49, no. 3, pp. 709–721, Jun. 2005.

[12] F. Azmat, Y. Chen, and N. Stocks, “Analysis of spectrum occupancy using machine learning algorithms,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 6853–6860, Sept 2016.

[13] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, “Machine learning paradigms for next-generation wireless networks,” *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, April 2017.

[14] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Proc. of Interspeech*, 2014.

[15] D. López, E. Rivas, and O. Gualdrón, “Primary user characterization for cognitive radio wireless networks using a neural system based on deep learning,” *Artif. Intell. Rev.*, Dec. 2017.

[16] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *arXiv preprint arXiv:1506.00019*, 2015.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 9, pp. 1735–1780, Nov. 1997.

[18] L. Yu, J. Chen, and G. Ding, “Spectrum prediction via long short term memory,” in *3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec 2017, pp. 643–647.

[19] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, “Deep learning models for wireless signal classification with distributed low-cost spectrum sensors,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018.

[20] L. Yu, J. Chen, G. Ding, Y. Tu, J. Yang, and J. Sun, “Spectrum prediction based on taguchi method in deep learning with long short-term memory,” *IEEE Access*, vol. 6, pp. 45 923–45 933, 2018.

[21] M. López-Benítez and F. Casadevall, “Improved energy detection spectrum sensing for cognitive radio,” *IET Commun.*, vol. 6, no. 8, pp. 785–796, 2012.