Applying Deep Neural Networks for Duty Cycle Estimation

Ahmed Al-Tahmeesschi[1], Kenta Umebayashi[1], Hiroki Iwata[1], Miguel López-Benítez[2, 3], and Janne Lehtomäki[4]

[1]Graduate School of Engineering, Tokyo University of Agriculture and Technology, Japan
[2]Dept. of Electrical Engineering and Electronics, University of Liverpool, United Kingdom
[3]ARIES Research Centre, Antonio de Nebrija University, Spain
[4]Centre for Wireless Communications, University of Oulu, Finland

***Abstract*—A pro-active spectrum usage prediction is a key technique in decision making and spectrum selection for dynamic spectrum access systems. This work focuses on the estimation of the duty cycle (DC) metric to reflect spectrum usage. The prediction is formulated as a time-series regression problem. Deep neural networks (DNNs) is selected to obtain accurate predictions of channel usage. Namely, Multilayer perceptron (MLP), Long short term memory (LSTM) and a hybrid model based on convolutional neural network followed by an LSTM (CNN-LSTM) layer are selected. The hyper-parameters selection has been optimised utilising both grid search and multi-stage grid search. Moreover, in many cases, the spectrum usage is measured on a smaller time scale from the actual required one. Hence, down-sampling and averaging is required. Averaging operation results in flattening the data and losing essential features to assist DNN to predict the channel usage. We show what is the minimum required time resolution to have a pro-active prediction system. Then, we propose utilising feature engineering to improve prediction accuracy. All the proposed DNNs approaches are trained on real-life measurements. The experimental evaluation demonstrated a high potential of DNNs to learn from previous spectrum usage and accurately predict the spectrum usage. Moreover, adding input features significantly assists the system to achieve accurate predictions in a pro-active manner.**

***Keywords*—** *Cognitive radio, deep neural networks, spectrum awareness, duty cycle.*

I. Introduction

The spectrum scarcity led to a significant interest in dynamic spectrum access principle [1, 2]. Several spectrum usage surveys have demonstrated the under-utilisation of the current spectrum allocation strategies. Having a dynamic access paradigm would highly increase spectrum efficiency [3]. In this scenario, the spectrum would be dynamically and autonomously assigned based on actual user usage and demands. However, this type of approach is highly dependent on accurate estimation of the temporal spectrum usage in terms of duty cycle (DC). DC is defined as the fraction of one period in which the channel is occupied.

Having a pro-active spectrum prediction system could assist the cognitive users by reducing the energy consumption in spectrum sensing [4] and to improve spectrum access and minimise interference on the primary network [5]. Recently, several attempts have been conducted to forecast the spectrum usage such as [6] and a comprehensive survey [7]. However, most of the conducted studies are based on conventional statistical techniques (a regression approach) such as autoregressive moving average (ARIMA) [8].

In the last few years, the topic of deep neural networks (DNN) has attracted significant attention in the field of wireless communications, such as in automatic modulation recognition [9], indoor localisation [10] and path loss exponent estimation in radio wave propagation [11]. [12] provides a comprehensive survey on DNN utilisation in smart wireless networks. This is mainly due to the advancement in massively parallel GPU architecture and high-level languages [13]. Motivated by the vast applicability of DNNs, we propose the utilisation of DNNs for spectrum usage prediction.

Even with the success of using DNNs in several areas, only a few studies considered utilising DNN in spectrum occupancy prediction. In [14], the channel occupancy in the form of a binary classification is considered. The closest to our proposed work is the one reported in [15]. Nevertheless, there are several differences between the reported and ours. First, different optimisation approaches are considered in both works. Second, a hybrid DNN is also considered in our work.

Another problem not been considered in previous works is when the dataset is down-sampled (i.e., several DC values are block averaged). The averaged DC blocks are then used to predict the next DC block. For instance, the measurements provide DC estimations over 100 msec time windows, but the network requirements predictions is over 5 seconds. In this case, having accurate DC estimations at larger time scales provide valuable information on which channel will be empty for a longer period of time, hence selection of channels with less data usage over long periods of time. Having DC estimation over longer periods improves spectrum allocation efficiency and reduces the number of channel hops for the dynamic spectrum access system users. First, we show the required block size (in time) to obtain an accurate estimation of the next step DC. Then, we propose to improve DC prediction by utilising input features beside the historical samples of DC, such as, samples' variance of the input dataset. Including these input features provide significant improvements to the estimation accuracy at larger DC block sizes.

This work aims to tackle the problem of downsampling by utilising DNN with suitable tuning. Then to further improve the prediction accuracy we introduce more input features to assist the DNN to make a pro-active estimation. The contributions of this work are outlined as follows:

1) Propose a spectrum usage prediction approach based on deep neural networks, namely MLP, LSTM and a

hybrid CNN-LSTM. It predicts the duty cycle based on experimental dataset.

2) The performance of DNN is highly influenced by hyper-parameters selection. Hence, the optimisation/tuning of the DNNs is accomplished by utilising grid search algorithm for the case of MLP and LSTM. A multi-layer grid search is implemented for CNN-LSTM. Extensive evaluation of the proposed methods is presented.
3) It is shown that the minimum average window size to accurately estimate the DC is 1 sec. This can be extended to 5 sec by applying feature engineering and extracting information from the input dataset to be used as features (such as, variance, slope) on the input data to further enhance the prediction accuracy.

The remainder of this paper is organised as follows. First, Section II presents the measurement system and the problem addressed in this work. Section III presents the considered models for time-series prediction with the evaluation metrics to assess the prediction performance. The hyper-parameters optimisation approaches are described in Section IV. Section V presents dataset preprocessing and preparations, as well as, data feature engineering. Section VI provide the simulation results for the considered prediction methods. Finally, Section VII summarises and concludes this work.

II. Measurement Setup and Methodology

A measurement system has been developed to continuously monitor the spectrum usage data as seen in Fig. 1. The system includes two real time spectrum analysers (RSAs), an external control trigger, a network attached storage (NAS), a measurement system control computer, a data analysis computer and a switching hub. The RSAs are configured to monitor the same channel, hence they operate in an alternating manner. The first RSA captures the IQ data while the second RSA transfers the captured IQ data to the network storage unit. The alternating functionality is controlled by the external trigger which is in turn controlled by the measurement system control computer. All data processing, such as detection and DC calculation are handled by the data analysis computer [16].

All spectrum usage measurements took place in WiFi channel 6 (centred at 2427MHz) with a sampling frequency of 1.75 MHz and bandwidth of 1.25 MHz. The first RSA will measure the IQ data over a time duration of T_s = 100 msec. Meanwhile, the second RSA will upload the previously captured IQ data to the network storage device. Then they swap operation (i.e., the first RSA uploads captured IQ and second RSA measures the IQ). By this alternating strategy, measurements without time domain gaps are achieved. As for signal detection, constant false alarm ratio (CFAR) strategy with probability of 0.01 is utilised. In order to find the detection threshold, one of the RSAs antennas was terminated and the noise floor was estimated based on dataset of 5 mins.

The measurement dataset spanned over 4 days, which provided 3456000 DCs sampled over 100 msec. It is important to notice that each DC value is based on large number of samples (received within 100 msec window). While it is possible to utilise an even smaller time duration (T_s) for the DC estimation, but this would result in having a larger dataset with a significant increase in DNN training complexity and larger storage requirement. Moreover, from the obtained results, it can be concluded that there is no need for higher time resolution as will be shown in Section VI.

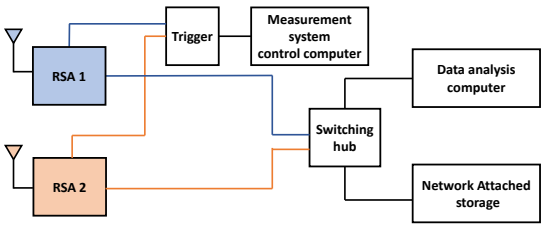


Fig. 1. Measurement system.

The input measurements consist of the temporally ordered spectrum usages Φ estimated at every 100 msec. In many cases, it is beneficial to obtain Φ estimation over longer periods. From a regression point of view, time resolution of input data should be equal to that of the values to be predicted. Hence a new spectrum usage (Φ_c) with estimation window of T_c can be defined as:

$$\Phi c = \frac{\sum_{k=1}^{K} \Phi_k}{K}, \tag{1}$$

where, $K = \frac{T_c}{T_s}$ and k is the DC index number. In Section VI, the estimation accuracy of different T_c values is studied.

III. Prediction Methods

In this section, first the considered models for time-series prediction are introduced, then the evaluation metric to asses the prediction performance of a DNN is defined.

A. MLP network

An MLP network is a feed-forward neural network based on the back propagation algorithm. An MLP consists of at least 3 fully connected (dense) layers namely, input layer, hidden layer(s) and output layer [17]. Each of the network layers includes a single or multiple neurons. The mathematical representation of a neural output is given as:

$$y_m = \psi\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2}$$

where w and x with different subscripts are the weights of transformation and input to neurons respectively and b is the bias. y_m is the neuron output. n is the number of inputs to a neuron and $\psi(.)$ is the non-linear activation function. The activation function is used to describe the non-linear properties between neuron input and output. In this work its assumed that $\psi(.)$ is a ReLU activation function which is defined as $\psi(z) = max(0, z)$. Using ReLU function in the hidden layers provides several advantages over other activation functions such as sigmoid or Tanh including the increase in training speed and reducing the likelihood of vanishing gradient [18].

B. LSTM network

An LSTM network is a special type of recurrent neural networks (RNNs). It was first proposed in [19] as an improvement over RNN. RNNs are usually used for time-series data type. Similar to any DNN type, an LSTM includes an input layer,

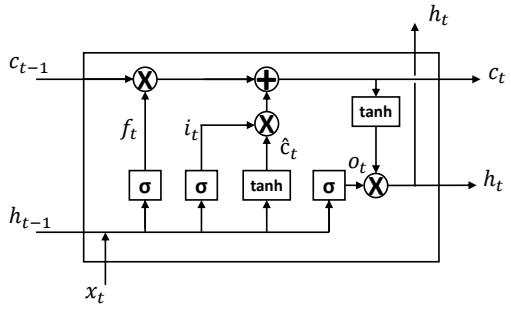


Fig. 2. LSTM cell.

hidden layer(s) and an output layer. LSTM was proposed to solve the problem of long term dependencies by adding an adaptive memory unit (cell state). The cell state unit value is only changed in a linear manner as can be seen in Fig. 2.

A standard LSTM layer includes three gates, an input gate (i_t), a forget gate (f_t) and an output gate (o_t). The input gate decides the amount of input x_t to the control unit c_t. The forget gate adjusts the value of the previous control unit c_{t-1}. The output gate controls the extent to which the value in memory is used to compute the output activation block. The gates are implemented with a sigmoid function which outputs a value between 0 and 1 to control information flow in an LSTM layer. The mathematical representation of the gates is given as:

$$i_t = \sigma\left(W_x^i \cdot x_t + W_h^i \cdot h_{t-1} + b_i\right), \tag{3}$$

$$f_t = \sigma\left(W_x^f \cdot x_t + W_h^f \cdot h_{t-1} + b_f\right), \tag{4}$$

$$o_t = \sigma\left(W_x^o \cdot x_t + W_h^o \cdot h_{t-1} + b_o\right), \tag{5}$$

where W_x, W_h and b are the input weights, recurrent weights and the biases in an LSTM cell, respectively. σ is the sigmoid function. x_t and h_{t-1} are the input and the preceding hidden state values, respectively. Before generating the hidden state c_t a temporary value $\hat{c}_t$ is generated first as follows:

$$\hat{c}_t = tanh\left(W_x^c \cdot x_t + W_h^c \cdot h_{i-1} + b_c\right), \tag{6}$$

then, the updated hidden state is obtained from:

$$c_t = i_t \odot \hat{c}_t + f_t \odot c_{t-1}, \tag{7}$$

where $\odot$ denotes the element-wise multiplication. Finally, the output of LSTM block can be expressed as:

$$h_t = o_t \odot tanh(c_t). \tag{8}$$

From the above expressions, it can be concluded that the gates play a vital role in controlling the historical information travelling in the LSTM networks.

C. Hybrid CNN-LSTM network

A hybrid CNN-LSTM is also considered in this work. The CNN part of the network consists of a single or several 1D convolutional layer(s) and a flatten layer. As the name implies, a convolutional layer performs convolution operation to amplify local features and produce the feature map. The convolutional layer utilises only part of the previous layer connections which makes it more efficient than a dense layer. The output of a convolutional layer is then fed to the flattened layer to transform it into one long vector that can then be used as input to the subsequent layers. The output of CNN network is fed to the subsequent LSTM network to learn long-range temporal dependencies. Adding a CNN layer on top of LSTM makes the DNN even more suitable for estimation. More details are presented in Section IV-B.

A pooling layer is not considered in this work as it involves downsampling for the output of convolutional layers, which could lead to loss of sequence characteristics. A similar approach can be found in [9]. A vanilla CNN is usually utilised for image like data [20], hence results for a stand-alone CNN are not considered in this work.

D. Performance evaluation metrics

In order to assess the suitability of the proposed models in predicting the DC, the R Squared (R^2) metric is considered [21]. The R^2 takes values of the range between $-\infty$ and 1 making it easier to interpret. Values of R^2 closer to 1 indicate that the model accounts for most of the variance in the dataset.

$$R^2 = 1 - \frac{\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{N}(y_k - \bar{y}_k)^2}, \tag{9}$$

where y_k and $\hat{y}_k$ are the actual and predicted DC values respectively. N is the number of predictions and $\bar{y}_k$ is the mean of the actual DCs.

IV. Hyper-parameters optimisation

The hyper-parameters design is an essential task in the design of DNNs. In this section, two hyper-parameters optimisation approaches are considered, grid search (GS) utilised for MLP and LSTM and multi-stage grid search applied for hybrid CNN-LSTM.

A. Grid search

The performance of a DNN is highly influenced by the selection of the hyper-parameters. In order to properly tune the DNN, an exhaustive grid-search is utilised to find the optimal hyper-parameters' values. Fig. (3), demonstrates the flowchart of the proposed GS algorithm. The architecture of MLP is utilised for explanation purposes. Nevertheless, the same concept is also applicable to LSTM networks. A total of 256 (4^4) possible combinations for each DNN (MLP and LSTM) model are searched through using GS. The following hyper-parameters are optimised:

1) The first hyper-parameter to optimise is the depth of the neural network (i.e., the number of hidden layers). The number of hidden layers is set to 1, 2, 3, and 4. Adding more layers increases the model ability to interpret inputs to outputs, but would result in overfitting with training dataset if too many layers were added.
2) The number of neurons or selecting the width of the neural network. In theory, a very wide neural network with a single hidden layer can obtain the same accuracy as a multi-layer deep neural network at the expense of increasing the complexity of training. In this work, the widths of 10, 20, 30 and 40 are considered for all hidden layers.
3) The activation function transforms the summed weighted inputs to the output. The following linear and non-linear activation functions are considered in this work: sigmoid,

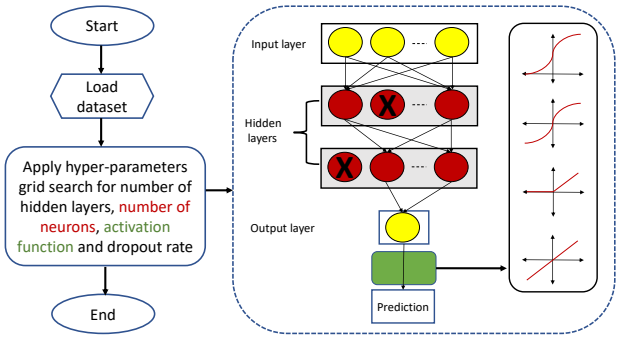


Fig. 3. Grid search flowchart.

TABLE I
HYPER-PARAMETERS SETTINGS FOR THE GRID SEARCH.

Hyper-parameter	Settings
Number of hidden layers	1, 2, 3, 4
Number of neurons	10, 20, 30, 40
Output activation function	Tanh, Linear, Sigmoid, ReLU
Drop out	1, 0.9, 0.75, 0.5
Batch size	128
Learning rate	0.001
Losses	MSE
Optimiser	Adam
Epochs	300

ReLU, Tanh and linear. The activation functions are applied to the third layer (output layer).

4) The last hyper-parameter is the dropout rate. Dropout is used as a regularisation method where some number of layer output is randomly ignored. The dropout is only applied on the hidden layers neurons'. The dropout rate is selected to have values of 1.0, 0.9, 0.75, and 0.5, where 1.0 means no drop out is considered.

Table I summarises the considered hyper-parameters for MLP and LSTM.

B. CNN-LSTM hyper-parameters selection

The grid search algorithm provides an optimal hyper-parameters optimisation, but suffer from high computational complexity. Implementing directly a grid search to optimise all the hyper-parameters in the hybrid CNN-LSTM would require a significant amount of time and resources. Inspired by the multi-stage deep model optimisation proposed in [22], a new optimisation stage for the CNN is added. The optimisation procedure would be to first optimise hyper-parameters of the LSTM layer, then the CNN.

The CNN hyper-parameters are optimised on top of the optimum LSTM layer (i.e. optimum values for hyper-parameters obtained from normal GS) as can be seen in Fig. 4. The following CNN hyper-parameters are optimised.

1) The number of convolution layers act as feature or pattern detectors. First layer detects large features, then second layer smaller features and so on. Adding many layers might result in over-fitting and saturation.
2) The number of filters decides how many filter weights are learned during backpropagation algorithm.
3) The filter size, usually a small number of filters is utilised. The value of filters are automatically decided based on the dataset and backpropagation algorithm.

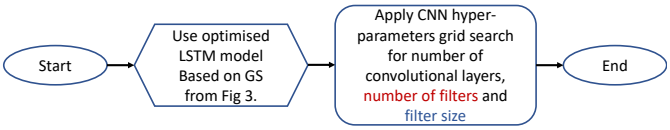


Fig. 4. Two stage grid search.

TABLE II
CNN LAYER HYPER-PARAMETERS SETTINGS FOR THE GRID SEARCH.

Hyper-parameter	Settings
Number of convolutional layers	1, 2, 3, 4
Number of filters	32, 64, 128
Kernel length	2, 3, 4
Pooling layer size	1, 0.9, 0.75, 0.5
Batch size	128
Learning rate	0.001
Losses	MSE
Optimiser	Adam
Epochs	300

A ReLU activation function is utilised for all of the convolutional layers. Table II summarises the considered hyper-parameters for CNN-LSTM.

V. DATASET PREPROCESSING

This section describes the dataset preparations for supervised DNN as well as features extraction for improved prediction accuracy.

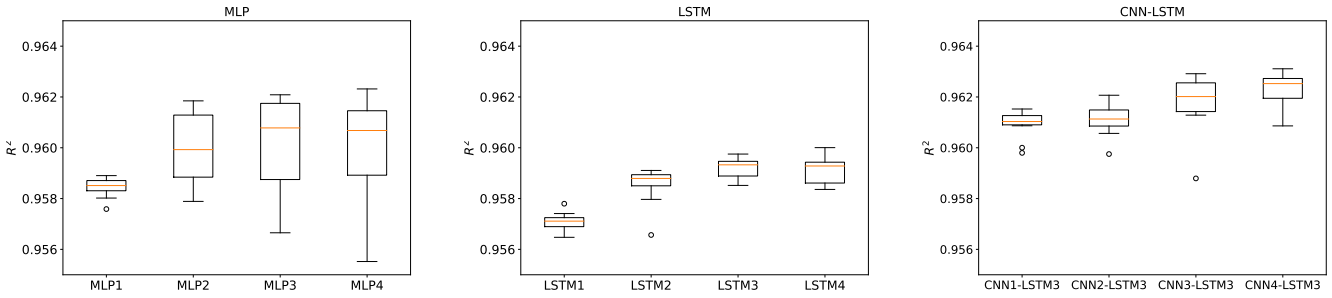
A. Walk forward validation

The supervised training dataset is made using sliding window validation approach. The dataset is divided into sliding windows. Each time step of the training dataset will be walked one step at a time (one step here is a single DC value). The sliding window size is set to the number of historical DC measurements. The walk forward could be thought of as in real-life scenario where at every time a spectrum measurement is done and used to forecast the following DC.

B. Testing for stationary

The advances in DNN methods made them rather independent on data preprocessing (such as scaling and differencing). This statement holds to a large extent for MLPs, but an LSTM for time-series forecasting might require some preprocessing to further improve it is performance. Mainly, the data should be tested for stationarity (i.e., the dataset should be time independent and maintain the same mean, variance and autocorrelation throughout the time). Stationarity is essential to improve the LSTM model prediction accuracy.

All of the re-sampled time series at different time instants need to be tested for stationarity. If the series were found to be nonstationary, then methods such as differencing must be included before the training phase. The augmented Dickey-Fuller (ADF) statistical test which is part of unit root test is used to confirm whether the datasets are stationary or non-stationary [23]. When the p value is above 0.05, the null hypothesis is failed to be rejected, hence, the data has unit root and thus none stationary. Otherwise, the dataset does not have a unit root and is thus stationary. After analysing our datasets, the dataset is found to be stationary for all time steps samples.



(a) (b) (c)

Fig. 5. Accuracy for multiple DNNs with $T_c = 1$ sec and only DC as input feature. (a) MLP, (b) LSTM, (c) CNN-LSTM.

C. Input features engineering

In many cases, the estimated spectrum usage window needs to be higher than the original dataset. In a time-series dataset, this means the original data set needs to be averaged then downsampled (i.e., block averaging). The problem becomes more significant as the window size increases (with both K and T_c being larger) as averaging would flatten the input dataset rendering a pro-active prediction significantly more complex.

To overcome this problem, we propose the inclusion of the following features besides the downsampled DC (Φ_c). Hence the input vector to the DNN will have the shape of $z \times p$, where z is the number of look back points and p is the number of input features. The considered input features are the input sample variance, slope and last DC value (i.e., Φ_K).

1) Observed variance of input samples. The idea of using the observed variance is to provide a measure of how much fluctuation is present in the downsampled dataset. It can be computed as: $Var(\Phi) = \frac{\sum_{k=1}^{K}(\Phi_k - \Phi_c)^2}{K}$.
2) Slope between the last two DC components. It provides an understanding if the Φ value was increasing or decreasing. It can be estimated as: $slope = \Phi_K - \Phi_{K-1}$.
3) The last DC component Φ_K (i.e., the last value before downsampling for the given time window).

VI. Experimental evaluation

In this work, Python 3 language is used alongside Keras [24] API with TensorFlow [25] as backend. Given that neural networks models are stochastic, hence, different weights will result at each training time even when the same model configuration is utilised. In order to address the model accuracy evaluation, each model configuration is evaluated multiple times (10 times in our case). The reported accuracy is averaged across the evaluations. A batch size of 128 is used for the training. The maximum number of epochs is set to 300. To ensure the DNNs do not overfit, an early stopping algorithm is utilised with an early stop of 20. At each prediction point, we assume that 5 previous DC measurements are present when making the prediction.

First, we would like to investigate the impact of adding more hidden layers on accuracy. Based on the optimised hyper-parameters for each of the hidden layers number, the boxplot of the best performing DNN models is shown in Fig. 5. The middle line inside each box represents the median accuracy value and the lower and upper edges represent the first and third quartiles, respectively. The end of whiskers shown on the lower and upper sides are the minimum and maximum values, respectively. The small circles are the outlier values.

Fig. 5(a) shows the performance of four MLP architectures with different hidden layers number. MLP1 means a single hidden layer, MLP2 means two hidden layers and so on. Sigmoid activation function was found to provide the best outcome with 40 neurons in each hidden layer and no dropout. As can be appreciated, MLP3 and MLP4 have similar performances with a slight advantage for MLP3 in terms of a slightly better median value and less complexity, hence will be considered in the rest of this work. This could be a result that all signal characteristics are being learned when having 3 hidden layers. Hence no actual benefit from adding extra layers. As for Fig. 5(b), shows LSTM network prediction accuracy. LSTM1, LSTM2, LSTM3, and LSTM4 refer for single, two, three and four hidden layers, respectively. Three hidden layers is found to provide the best performance in terms of median values. Interestingly, even though LSTM is considered as more suitable for time-series type of data, it provided less accuracy than MLP3. The LSTM gave smaller median value in comparison with the MLP but with smaller variance which means more consistent results when compared with MLP. The accuracy of the hybrid CNN-LSTM is shown in Fig. 5(c). As can be appreciated, the hybrid CNN4-LSTM3 (CNN4 stands for 4 convolutional layers) gave the best accuracy when compared to other CNN-LSTMs, stand-alone MLP3 and LSTM3. It can be concluded that using a hybrid CNN-LSTM slightly improves the prediction accuracy.

As can be appreciated from Fig. 6, when utilising a small averaging window of $T_c = 1$ sec the accuracy of prediction is high and the DNN models can predict the actual DC with high accuracy and without having time delayed predictions. Unfortunately, this observation does not hold when $T_c = 5$ sec. As can be seen in Fig. 7, all the DNNs provide an inaccurate predictions. This indicates that the input spectrum usage data is losing some essential features, rendering it difficult to have a pro-active estimation of future spectrum utilisation.

Table III shows the prediction accuracy for the proposed DNN models as a function of time and input features. In general, the hybrid CNN4-LSTM3 provides improvement in accuracy over MLP3 and LSTM3 depending on T_c value. Nevertheless, when $T_c > 1$ sec, the prediction accuracy degrades significantly for all the proposed DNNs. Using DC as the only input feature is insufficient as long as the DC estimation is no longer than 1 sec. This means the larger

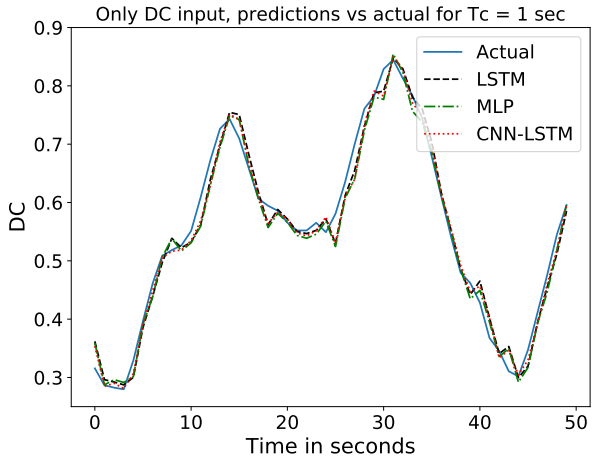


Fig. 6. Accuracy for $T_c = 1$ sec and only DC as input feature.

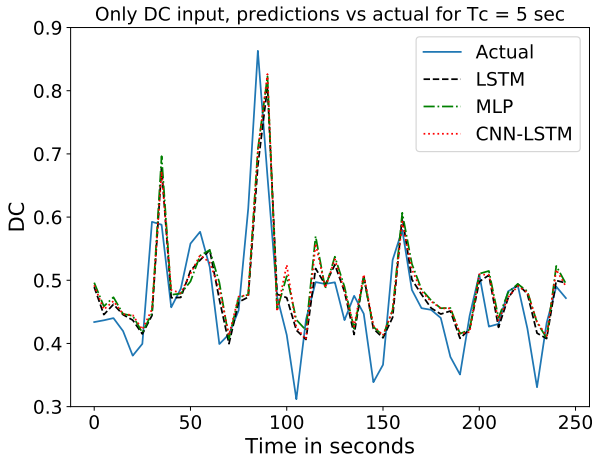


Fig. 7. Accuracy for $T_c = 5$ sec and only DC as input feature.

the averaging window, the more features are being lost in the downsampling phase which makes it more difficult for the DNN to estimate the next DC value. Thus, we propose the addition of more features to the input layer to have an accurate representation of the input sequence. Hence, a more robust trained DNN.

The third columns set under "DC+Var" of Table III show when both DC and sample variance are used together for the estimation. Both of the last value and slope have significant importance at improving the estimation accuracy. Another important observation is that MLP, which is the least complex model, provided decent results. In fact, MLP has surpassed LSTM nearly in most scenarios and provided more accurate predictions. The CNN-LSTM provided a minor improvement over MLP. The results suggest that the most important feature is when the last DC value Φ_K is available, which provided most of the improvement, while when using all the input features further improvement can be noticed.

Figs. 8 and 9 show the prediction accuracy when DC with last value Φ_K and all input features are used, respectively. While the first case improves the prediction over when only having DC historical information. Nevertheless, Having all features further improve the prediction accuracy.

Finally, Table III shows that both MLP and CNN-LSTM have comparable results. Hence we provide the probability density function (PDF) of the residuals (actual - prediction) is plotted in Fig. 10. As it can be appreciated positive means model is underestimating the actual values and minus means overestimating. Ideally, the best performance achieved when the PDF has a peak at residuals = 0.0 and the PDF is zero everywhere else. In case of spectrum assignment and selection, it is preferred to have an over estimation of the actual spectrum utilisation than underestimation. As protecting the primary user from interference prevails over secondary network performance. Hence the hybrid CNN-LSTM provides a better estimation of DC.

Future work will include further improving the estimation of DC through larger time scales averaging windows T_c, besides studying the optimum number of delays that are required for the input.

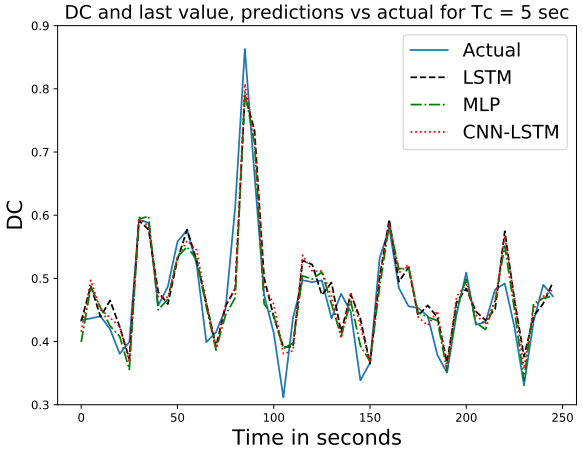


Fig. 8. Accuracy for $T_c = 5$ sec and DC and Φ_k as input features.

VII. Conclusions

A pro-active spectrum usage estimator is essential for dynamic access systems. In this work, the feasibility of using DNNs for pro-active spectrum prediction is investigated. Several DNN models namely, MLP, LSTM and hybrid CNN-LSTM have been proposed for the problem of spectrum occupancy prediction. It is shown that for WiFi systems it is possible to obtain accurate DC estimation when using a window duration of 1 second with only DC as input feature. Unfortunately, when increasing the DC estimation window the accuracy of DNNs reduced rapidly. To solve this, a solution is proposed based on utilising more input signal features (such as, variance and slope) to assist the DNN to learn useful features on the input time-series dataset. Simulation results demonstrated high potential of DNNs to learn from previous spectrum usage and accurately predict the spectrum usage especially when adding multiple input features and hence increasing the prediction time scale (T_c).

Acknowledgement

This work was supported by the European Commission in the framework of the H2020-EUJ-02-2018 project 5G-Enhance (Grant agreement no. 815056) and the Ministry

TABLE III
ACCURACY OF PREDICTIONS, T_c IS GIVEN IN MSEC.

Tc	DC			DC + Var			DC + Slope			DC + Last			All		
	MLP	LSTM	C-LSTM	MLP	LSTM	C-LSTM	MLP	LSTM	C-LSTM	MLP	LSTM	C-LSTM	MLP	LSTM	C-LSTM
0.5	0.992	0.992	0.995	0.992	0.992	0.995	0.997	0.997	0.998	0.996	0.997	0.997	0.997	0.997	0.997
1	0.96	0.962	0.966	0.96	0.962	0.968	0.986	9885	0.989	0.987	0.987	0.991	0.989	0.99	0.993
3	0.777	0.785	0.786	0.807	0.791	0.827	0.797	0.788	0.828	0.922	0.924	0.928	0.938	0.934	0.939
5	0.634	0.643	0.650	0.668	0.650	0.678	0.783	0.781	0.788	0.848	0.844	0.847	0.871	0.858	0.881
8	0.347	0.329	0.351	0.4	0.351	0.453	0.499	0.517	0.526	0.723	0.714	0.731	0.749	0.734	0.749
10	0.263	0.241	0.267	0.309	0.231	0.323	0.411	0.369	0.418	0.653	0.588	0.667	0.684	0.665	0.697

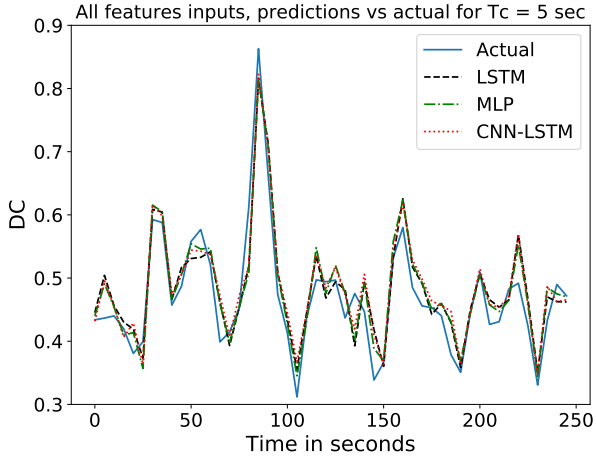


Fig. 9. Accuracy for T_c = 5 sec and all input features.

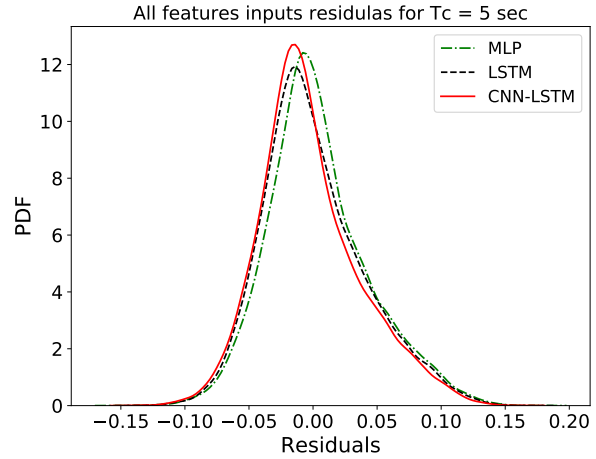


Fig. 10. Residuals density for T_c = 5 sec and all input features.

of Internal Affairs and Communications (MIC) Japan. The work of M. López-Benítez was supported by British Council under UKIERI DST Thematic Partnerships 2016-17 (ref. DST-198/2017). The work of J. Lehtomäki was supported by the Academy of Finland 6Genesis Flagship (grant no. 318927).

REFERENCES

[1] C. H. Liu, P. Pawelczak, and D. Cabric, "Primary user traffic classification in dynamic spectrum access networks," *IEEE Journal on Selected Areas in Comms.*, vol. 32, no. 11, pp. 2237–2251, November 2014.

[2] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006.

[3] T. Fujii and K. Umebayashi, "Smart spectrum for future wireless world," *IEICE Transactions on Communications*, vol. E100B, no. 9, pp. 1661–1673, Sep. 2017.

[4] X. Xing, T. Jing, Y. Huo, H. Li, and X. Cheng, "Channel quality prediction based on bayesian inference in cognitive radio networks," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 1465–1473.

[5] P. Zuo, X. Wang, W. Linghu, R. Sun, T. Peng, and W. Wang, "Prediction-based spectrum access optimization in cognitive radio networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–7.

[6] K. E. Baddour, A. Ghasemi, and H. Rutagemwa, "Spectrum occupancy prediction for land mobile radio bands using a recommender system," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug 2018, pp. 1–6.

[7] G. Ding, Y. Jiao, J. Wang, Y. Zou, Q. Wu, Y. Yao, and L. Hanzo, "Spectrum inference in cognitive radio networks: Algorithms and applications," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 150–182, Firstquarter 2018.

[8] Z. Wang and S. Salous, "Spectrum occupancy statistics and time series models for cognitive radio," *Journal of Signal Processing Systems*, vol. 62, no. 2, pp. 145–155, Feb 2011.

[9] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4074–4077, Apr 2019.

[10] S. Liu, R. Y. Chang, and F. Chien, "Analysis and visualization of deep neural networks in device-free wi-fi indoor localization," *IEEE Access*, vol. 7, pp. 69 379–69 392, 2019.

[11] H. F. Ates, S. M. Hashir, T. Baykas, and B. K. Gunturk, "Path loss exponent and shadowing factor prediction from satellite images using deep learning," *IEEE Access*, vol. 7, pp. 101 366–101 375, 2019.

[12] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2595–2621, Fourthquarter 2018.

[13] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec 2017.

[14] Lixing Yu, Qianlong Wang, Yifan Guo, and Pan Li, "Spectrum availability prediction in cognitive aerospace communications: A deep learning perspective," in *2017 Cognitive Communications for Aerospace Applications Workshop (CCAA)*, June 2017, pp. 1–4.

[15] 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.

[16] H. Iwata, K. Umebayashi, J. Lehtomäki, M. López-Benítez, and S. Narieda, "Development of a smart spectrum access prototype," in *26th European Conference on Networks and Communications (EuCNC 2017)*, Jul 2017, pp. 1–2.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[21] A. C. Cameron and F. A. Windmeijer, "An r-squared measure of goodness of fit for some common nonlinear regression models," *Journal of Econometrics*, vol. 77, no. 2, pp. 329 – 342, 1997.

[22] A. Tahmassebi, A. H. Gandomi, S. Fong, A. Meyer-Baese, and S. Y. Foo, "Multi-stage optimization of a deep model: A case study on ground motion modeling," *PLOS ONE*, vol. 13, no. 9, pp. 1–23, 09 2018.

[23] W. Enders, *Applied econometric time series*. Wiley, 2015.

[24] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, and et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.