Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Feature-Based Deep Neural Networks for Short-Term Prediction of WiFi Channel Occupancy Rate

AHMED AL-TAHMEESSCHI[1], **(Member, IEEE)**, KENTA UMEBAYASHI[1], **(Member, IEEE)**, HIROKI IWATA[1], JANNE LEHTOMÄKI[2], **(Member, IEEE)**, and MIGUEL LÓPEZ-BENÍTEZ[3,4], **(Senior Member, IEEE)**

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

Corresponding author: Ahmed Al-Tahmeesschi (e-mail: ahmedaa@go.tuat.ac.jp).

This work was supported by the European Commission in the framework of the H2020-EUJ-02-2018 project 5GEnhance (Grant agreement no. 815056), by "Strategic Information and Communications R&D Promotion Programme (SCOPE)" of Ministry of Internal Affairs and Communications (MIC) of Japan (Grant no. JPJ000595). The work of J. Lehtomäki was supported by the Academy of Finland 6Genesis Flagship (grant no. 318927). The work of M. López-Benítez was supported by British Council under UKIERI DST Thematic Partnerships 2016-17 (ref. DST-198/2017).

ABSTRACT

Spectrum occupancy prediction is a key enabling technology to facilitate a proactive resource allocation for dynamic spectrum management systems. This work focuses on the prediction of duty cycle (DC) metric that reflects spectrum usage (in the time domain). The spectrum usage is typically measured on a shorter time scale than needed for prediction. Hence, data thinning is required and we apply block averaging. However, averaging operation results in flattening the DC data and losing essential features to assist deep neural network (DNN) to predict the spectrum usage. To improve DC prediction after block averaging, a feature-based deep learning framework is proposed. Namely, long short-term memory (LSTM) and gated recurrent unit (GRU) are selected and enhanced by using features of the data, such as the variance of DC data in addition to DC data themself. The proposed model is capable of proactively predicting the spectrum usage by capturing complex relationships among various input features for the measured spectrum, thus providing higher prediction accuracy with an average improvement of 5% in RMSE compared with traditional models. Moreover, to have a better understanding of the proposed model, we quantify the effect of input features on the predicted spectrum usage values. Based on the most significant input features, a simpler and more efficient model is proposed to estimate DC with similar accuracy to when using all features.

INDEX TERMS 5G, Deep Neural Networks, Explainable AI, GRU, LSTM, Occupancy Rate, SHAP, Short-Term Prediction, Spectrum Awareness, WiFi.

I. INTRODUCTION

Unlike previous generations in wireless communications, the next-generation wireless networks are expected to be fast and capable of connecting several billions of devices. According to [1], the global mobile data traffic reached around 33EB per month in 2019 with the expectation that it will grow to 164EB per month by 2025.

This increase in both user traffic and number of users is accommodated with highly specialised use cases. In 5G, three distinct use cases are defined by 3rd Generation Partnership (3GPP), namely enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and massive Machine Type Communication (mMTC). Next-generation communication systems should be able to handle extreme use cases that do not fall under the three 3GPP categories as can be seen in Fig. 1.

Several spectrum usage surveys have shown the under-utilisation of current spectrum allocation strategies. Using a dynamic access paradigm would increase spectrum efficiency [2]. In this case, the spectrum would be autonomously as-

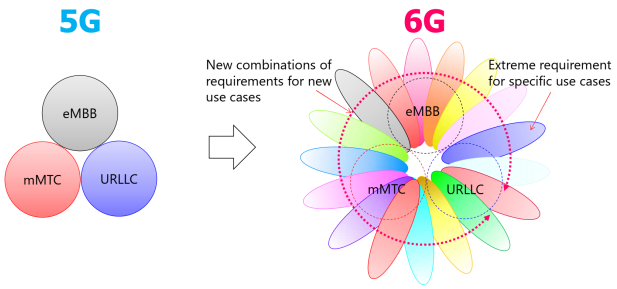


FIGURE 1. Image of technological development toward 6G [15].

signed based on actual user usage and demands. On the other hand, this type of approach is highly dependent on accurate prediction of the temporal spectrum usage in terms of duty cycle (DC). DC (also known as occupancy rate) is defined as the fraction of one period in which the channel is occupied.

In order to efficiently allocate network resources, it is necessary to proactively predict the network traffic instead of passively responding as in traditional approaches [3]. Thus, more advanced network infrastructure is essential to facilitate proactive spectrum resource allocation.

The problem of traffic prediction is a challenging task. As several factors could impact the network traffic, including the number of active users, time and location. Traditionally, time series temporal analysis and forecasting techniques are applied to estimate network traffic. However, most of the conducted studies are based on conventional statistical techniques (a regression approach) such as autoregressive moving average (ARIMA) [4], fractional ARIMA [5] or seasonal autoregressive moving average (SARIMA) [6].

Luckily, the advent of artificial intelligence (AI) can help network operators to automatically and efficiently adjust the network [7]. AI has been utilised recently for numerous mobile and wireless communications applications [8], including in automatic modulation recognition [9], indoor localisation [10] and path loss exponent estimation in radio wave propagation [11]. As for next-generation network management, it is expected that AI will have an impact on key areas, such as enhanced service quality, higher network efficiency, and improved network security. For instance, AI could be used to detect anomalies in network traffic by identifying unusual spectrum usages [12]. A comprehensive survey on deep neural network (DNN) utilisation in smart wireless networks is presented in [13]. This increase in using AI is mainly due to the advancement in massively parallel GPU architecture and high-level languages [14].

This work concerns with the prediction of DC. Specifically, this work focuses on the problem of short-term prediction of DC (i.e., DC is predicted over periods of tens of seconds). In many cases, the spectrum usage is measured on a shorter time scale than the actual required one. Hence, some kind of data thinning (or data decimation) is required and we apply block averaging to achieve it due to its simplicity. However, averaging operation results in flattening the data and losing essential features to assist DNN to predict the channel usage. The averaged DC blocks are then used to predict the next DC block. For instance, the measurements may provide DC estimations over 100 millisecond time window, but the network requires predictions that are over 30 seconds. In this case, having accurate DC predictions at larger time scales provide valuable information on which channel will be empty for a longer period of time, hence the selection of channels with less data usage over longer time periods. Having DC estimation over longer periods improves spectrum allocation efficiency and reduces the number of channel hops for the dynamic spectrum access system users. A feature-based solution with either long short-term memory (LSTM) or gated recurrent unit (GRU) models is proposed to improve DC prediction. The proposed method incorporates various features of DC data as well as original DC data itself as DNN inputs to improve the DC prediction. Notice that the proposed LSTM/GRU models differentiate from the typical LSTM/GRU models commonly employed in the literature in that they incorporate and exploit features extracted from the DC dataset to improve the prediction accuracy, while much of the literature does not include the addition of extra features from the input dataset and focuses mainly on changing the DNN model (for example using many hidden layers or using different deep learning algorithms [16, 17]). Adding several input features is a subtle but important difference that significantly improves the prediction accuracy. Moreover, this also constitutes a simpler, more practical, more convenient and also more effective approach than modifying the architecture of the employed DNN model as typically done in the literature.

When using several features at the input, an essential question arises, what is the effect of each feature at the output, which features are important. A solution based on SHAP values (SHapley Additive exPlanations) [18] is proposed to show the importance of each input feature. We apply SHAP framework to interpret the proposed DNN models to identify the most influential features contributing to the model's predictions while also quantifying their contribution level for individual predictions. Based on the obtained importance values, an efficient deep learning solution based on limiting the number of input features and only using the most significant ones is proposed.

The contributions of this work are outlined as follows:

1) In depth investigate, compare and evaluate the forecast of DC time series data on a number of solo forecasting approaches including ARIMA, seasonal ARIMA (SARIMA) and deep learning approaches such as multilayer perceptron (MLP), LSTM and GRU.
2) Investigate the prediction accuracy of original DNN models for the DC prediction when only DC is available at the input. Moreover, we show that DNNs have better learning capability when being exposed to different types of features at the input combined with first difference of the temporal series data to improve the prediction accuracy.

3) Utilise visualisation technology SHAP values to compare the significance of input features and their impact on the output of DNN model. Based on the most significant features an efficient DNN model is proposed.

The remainder of this paper is organised as follows. First, Section II provides literature survey for related traffic prediction and explainable AI methods. Section III presents the measurement system and the problem addressed in this work. Section IV presents the considered models for time series prediction with the evaluation metrics to assess the prediction performance. The hyper-parameters optimisation approach is described in Section V. Section VI presents dataset preprocessing and features extractions. Section VII provides the simulation results for the considered prediction methods. Finally, Section VIII summarises and concludes this work.

II. LITERATURE REVIEW

A. TRAFFIC PREDICTION

Wireless traffic prediction will have a significant impact on improving next-generation wireless networks. In the literature, several studies have been conducted to forecast the traffic occupancy. Majority of the models focus on predicting the network throughput, such as [17], [19], [20] and [21]. While in this work spectral occupancy rate is considered instead of data throughput. For dynamic spectrum access systems, spectral occupancy is a more important metric to consider, since it directly relates to the available spectrum resources. The relationship between traffic and spectrum occupancy is a complex one and no one-to-one mapping is possible. Therefore, it is crucial to specifically focus on forecasting spectrum occupancy.

Recently, several attempts have been conducted to forecast spectrum usage [22]. In [23], the authors concluded that for the land mobile radio bands there is no universally best statistical or machine learning method to predict spectrum occupancy rate. Thus, a recommender unit based on machine learning is used to select the best approach to predict spectrum occupancy. In [24], the channel occupancy in the form of binary classification is considered. This is different from our work, as we try to predict the value of occupancy rate rather than if the channel was busy or idle. In [16], the authors predict the spectrum availability following two approaches, classification and regression. In the classification scenario, the spectrum band is predicted as either idle or busy. While in the regression case, the received signal spectral density is predicted, which is also different from our work, as we try to predict the value of occupancy rate rather than the received signal power.

For the time series prediction problem, several studies concluded that using only traffic information as input feature is not sufficient, in fact, in [25] it was shown that using only traffic as input feature for enterprise network traffic prediction did not provide any advantage over traditional linear regression methods such as autoregressive integrated moving average (ARIMA). A solution based on utilising spatial input of several access points was used to improve the network traffic prediction. Similar observations were reported in [26] and [27]. In [28], statistical input features derived from the data itself were used to further improve the prediction accuracy.

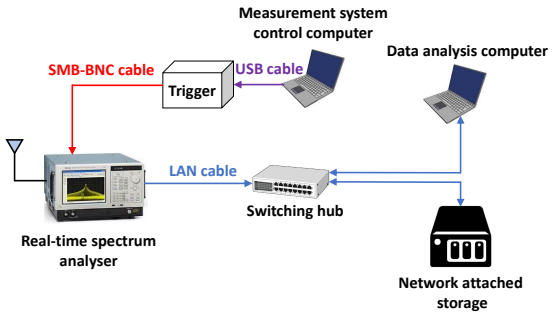


FIGURE 2. Measurement system.

Thus, it can be concluded that having more input features to the neural network such as spatial details of transmitters and users, number of active users, type of device used for internet or network access and other statistical features such as the ones described in [29] will impact the predicted traffic usage. Therefore, we would like to further study the importance of having more input features on the prediction accuracy for spectral occupancy. Another question that needs an answer is which features are important, and how much do the input features contribute to the forecasting outcome. The DNN complexity gets higher as the number of input features increases. This research aims to study these open questions.

B. EXPLAINABLE AI

One notable shortcoming of DNN models is finding a solid justification for their output. This imposes obstacles to their large-scale implementation [30]. Therefore, several studies have been interested in explainable artificial intelligence (XAI) to improve human ability to understand the process decisions made by DNNs. The goal is to find the impact of different input features on the prediction result. In the literature, several methods have been proposed [31], [32], [33] and [34].

One approach is to utilise SHAP values [18]. It breaks down a prediction to show the impact of each feature on the output. In the literature, SHAP values have been used to link input features with the model output. In [35], SHAP is used to improve the detection of adversarial attacks. The significance of temporal features for land mobile radio bands using machine learning model was considered in [23] and for LTE traffic using deep learning model in [36].

In this work, we are going to derive feature importance for DC prediction using deep learning models and SHAP values. Based on the most significant features an efficient DNN model is proposed.

III. MEASUREMENT SETUP AND METHODOLOGY

Spectrum usage measurements took place in WiFi channel 6 (centered at 2427MHz) with a sampling frequency of 20 MHz inside the TUAT university office (indoor environment). The measurement system included a real-time spectrum analyser (RSA), an external control trigger, a network-attached storage, a measurement system control computer, a data analysis computer and a switching hub. The start of capture time is controlled by an external trigger which is in turn controlled by the measurement system control computer. All data processing, such as detection and DC calculation, is handled by the data analysis computer. The measurement system is shown in Fig. 2. Measurements were recorded for working days only (i.e. no weekends or holidays WiFi traffic was recorded). The measurement did not occur in a continuous manner, as only working days are considered with a total of 13 days during February 2020.

DC is measured over a time duration of $T_s = 200$ ms. As for signal detection, a constant false alarm ratio (CFAR) strategy with a false alarm probability of 0.01 is utilised. In order to find the detection threshold, the RSA's antenna was terminated and the noise floor was estimated based on a dataset of 1 hour. The measurement dataset spanned over 13 days, which provided 5616000 DCs sampled over 200 ms. It is important to notice that each DC value is based on a large number of samples (received within 200 ms window). 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.

The input measurements consist of the temporally ordered spectrum usages Φ (i.e., DC values) estimated every 200 ms. In many cases, it is beneficial to obtain Φ estimation over longer periods. From a regression point of view, the 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, $K = \dfrac{T_c}{T_s}$ and k is the DC index number. In this work, T_c is selected to be an integer multiple of T_s. In Section VII, the estimation accuracy of different T_c values is studied.

In order to have a deeper understanding of the measured dataset. Figs. 3 and 4 show the occupancy rate measurements and their probability mass function (PMF), respectively for DC aggregated over $T_c = 30$ seconds.

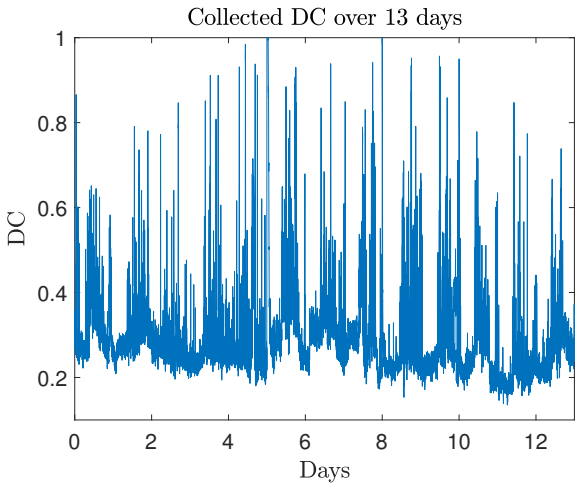


FIGURE 3. Measured occupancy rate.

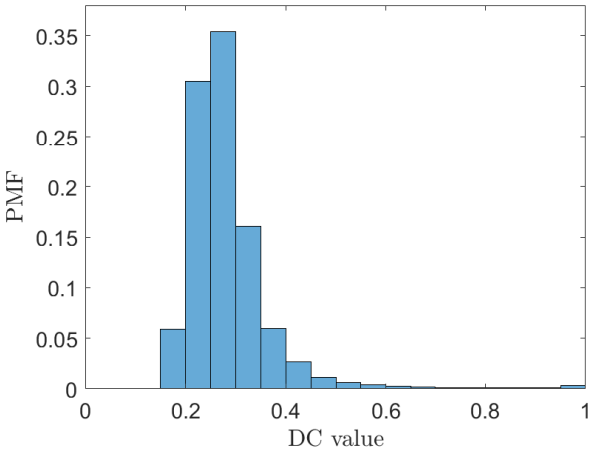


FIGURE 4. PMF of the measured occupancy rate.

IV. PREDICTION METHODS

In this section, the considered prediction techniques are described.

A. RANDOM WALK / NAIVE PREDICTOR

The random walk (RW) model is one of the simplest forecasting models. It assumes that every point y_t in the time series takes an independently and identically distributed value (e_t) step away from the previous point [37].

$$y_t = y_{t-1} + e_t. \tag{2}$$

In other words, the future value is equally likely to be higher or lower than the current one. Thus, RW model predicts that the future value will be equal to the last observed value. RW model is considered in this work for comparison purposes and to show that in specific scenarios more complex models could perform worse than a simple RW model.

B. ARIMA

The ARIMA model introduced by [38], is a flexible time series forecast method. It predicts the future values based on past observations (a linear function of past observations) and an error. ARIMA describes the time series by three fundamental aspects

1) Autoregressive terms (AR), the future value (forecasted) depends on weighted time-lagged values of itself.

$$y_t = \lambda_1 y_{t-1} + \lambda_2 y_{t-2} + \cdots + \lambda_p y_{t-p}, \quad (3)$$

where λ_j represents the AR coefficients and p is the number of previous observations.

2) Integrated terms (I), considered to make the time series stationary.

$$y_t = y_t - y_{t-1} - \cdots + y_{t-d}, \quad (4)$$

where d is the order of difference.

3) Moving average terms (MA), regression against past errors.

$$y_t = \Theta_1 \varepsilon_{t-1} + \Theta_2 \varepsilon_{t-2} + \cdots + \Theta_p \varepsilon_{t-q}, \quad (5)$$

where Θ_j represents the MA coefficients and q is the number of previous observations. ε is the residuals from fitting ARIMA model.

Utilising the backward shift operator ($L^k y_t = y_{t-k}$), the ARIMA model can be defined as:

$$(1 - \sum_{j=1}^{p} \lambda_j L^j)(1 - L)^d y_t = c + (1 + \sum_{i=1}^{q} \Theta_i L^i)\varepsilon_t, \quad (6)$$

where c is the constant in ARIMA model. The identification of p and q is from auto-correlation function (ACF) and partial auto-correlation function (PACF). It was found that ARIMA$(4, 1, 1)$ provided the best fit.

The general ARIMA model can be extended to incorporate the seasonal (SARIMA) variations in the time series. SARIMA can be expressed as SARIMA$(p, d, q)(P, D, Q)_s$, where P, D and Q are the number of seasonal AR terms, seasonal difference and the number of seasonal MA terms. The periodicity/seasonality is set by s.

Generally, d+D is equal to or smaller than 2 [39]. In our case, it was found that SARIMA$(4, 1, 1)(1, 1, 1)_s$ provided the best fit. s takes 1440 when predicting DC over 1 min (as there are 1440 minutes per day, hence $s = 1440$). Thus when forecasting for 1 min, SARIMA model becomes SARIMA$(4, 1, 1)(1, 1, 1)_{1440}$.

C. MLP

An MLP network is a feed-forward neural network based on the backpropagation algorithm. An MLP consists of at least 3 fully connected (dense) layers namely, input layer, hidden layer(s) and output layer [40]. Each of the network layers includes a single or multiple neurons. The mathematical representation of a neural output is given as:

$$y_m = \psi\left(\sum_{i=1}^{n} w_i x_i + b\right), \quad (7)$$

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 it is 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 [41].

D. LSTM

The LSTM network is a variation of recurrent neural networks (RNNs) which is typically used for time series data types. It was first proposed in [42] as an improvement over RNN to solve long-term dependency. The LSTM includes an input layer, 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. 5.

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. An output value of 0 means no input passing through the gate, where an output of 1 means all the input is passing through the gate. The mathematical representation of the gates is given as:

$$\text{Input gate} : i_t = \sigma\left(W_x^i \cdot x_t + W_h^i \cdot h_{t-1} + b_i\right), \quad (8)$$

$$\text{Forget gate} : f_t = \sigma\left(W_x^f \cdot x_t + W_h^f \cdot h_{t-1} + b_f\right), \quad (9)$$

$$\text{Output gate} : o_t = \sigma\left(W_x^o \cdot x_t + W_h^o \cdot h_{t-1} + b_o\right), \quad (10)$$

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 cell state values, respectively. Before generating the hidden cell state c_t a temporary value $\hat{c}_t$ is generated first as follows:

$$\text{Temporary cell state} : \hat{c}_t = tanh\left(W_x^c \cdot x_t + W_h^c \cdot h_{i-1} + b_c\right), \quad (11)$$

where tanh is the hyperbolic tangent. The updated hidden state is obtained from:

$$\text{Cell state} : c_t = i_t \odot \hat{c}_t + f_t \odot c_{t-1}, \quad (12)$$

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

$$\text{LSTM block output} : h_t = o_t \odot tanh(c_t). \quad (13)$$

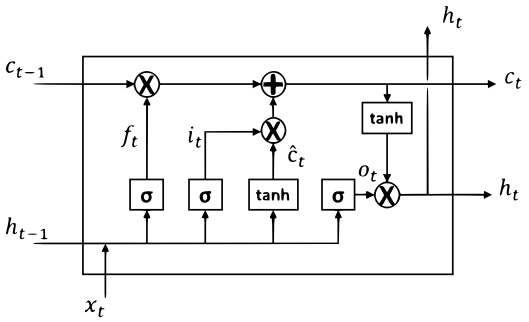


FIGURE 5. LSTM cell.

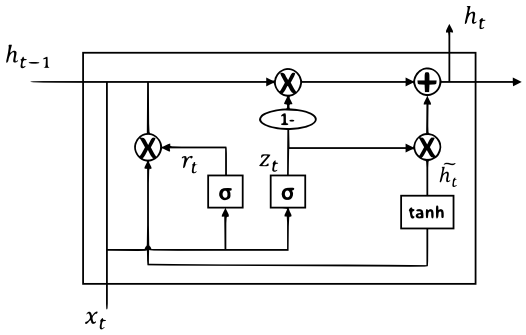


FIGURE 6. GRU cell.

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.

E. GATED RECURRENT UNIT (GRU)

The GRU is another RNN variant [43]. The main difference between GRU and LSTM is the GRU has only two gates, reset and update gates whereas an LSTM has three gates (namely input, output and forget gates). A GRU cell controls information flow similar to an LSTM cell, but without having to use a memory unit. It exposes the full hidden content without any control.

For a time series dataset prediction, GRU has comparable performance to LSTM but it is computationally more efficient (has a less complex structure). The GRU structure can be seen in Fig. 6 and the gate parameters r_t and z_t are given as:

$$\text{Reset gate}: r_t = \sigma\left(W_h^r \cdot h_{t-1} + W_{rx} \cdot x_t + b_r\right), \quad (14)$$

$$\text{Update gate}: z_t = \sigma\left(W_{zh} \cdot h_{t-1} + W_{zx} \cdot x_t + b_z\right), \quad (15)$$

where, σ is the sigmoid activation function.

$$\text{Memory}: \hat{h}_t = tanh\left(W_{hh}(r_t \odot h_{i-1}) + W_{hx} \cdot x_t + b_h\right), \quad (16)$$

The output of the GRU block can be expressed as:

$$\text{Final output}: h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t, \quad (17)$$

F. PERFORMANCE EVALUATION METRICS

In order to assess the suitability of the proposed models in predicting the DC, several metrics to measure the forecasting accuracy are considered. One of the most popular metrics is the root mean square error (RMSE) [44]. It can be defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}, \quad (18)$$

where y_k and $\hat{y}_k$ are the actual and predicted DC values, respectively. N is the number of predictions. The RMSE provides a higher weight for large errors because of the square term. This makes it more appealing for applications where large errors are not desirable.

Another important considered metric is the R Squared (R^2) metric or the coefficient of determination [45]. 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.

$$R^2 = 1 - \frac{\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{N}(y_k - \bar{y}_k)^2}, \quad (19)$$

where $\bar{y}_k$ is the mean of the actual DCs.

The mean average percentage error (MAPE) [46], provides an accuracy measure in terms of relative error. It is given by:

$$MAPE = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{y_k - \hat{y}_k}{y_k}\right|, \quad (20)$$

In this work, the RMSE, R^2 and MAPE metrics will be used to assess the performance of prediction algorithms.

V. HYPER-PARAMETERS OPTIMISATION

The hyper-parameters selection is an essential task in the design of DNNs. But the selection of optimum hyper-parameters is typically not possible [47], thus the grid-search (GS) approach is used to find the optimum parameters. In this Section, we consider the optimisation grid search utilised for MLP and LSTM neural networks.

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. 7, 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 model are searched thoroughly 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's ability to interpret inputs to outputs, but would result in overfitting with the 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, 30, 50 and 80 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, ReLU, tanh and linear which are applied to the third layer (output layer). As for the case of hidden layers, the activation function for MLP is selected to be always ReLU and for LSTM and GRU, a tanh activation function is selected.
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 to neurons of the hidden layer. The dropout rate is selected to have values of 1.0, 0.9, 0.75, and 0.5, where 1.0 means no dropout is considered.

Table 1 summarises the considered hyper-parameters for LSTM. The reason for selecting the MSE loss function is that the MSE loss function is used to ensure the trained model has no outlier predictions with large errors. As can be appreciated from Fig. 4, the DC distribution is highly skewed where the high DC values (between 0.6 and 1) have a small probability of occurrence and should be considered as outliers. In resource allocation, high DC values are important (since these moments are those where a more proactive resource allocation is required). Therefore, errors coming from these outliers should be weighted more (a larger error will provide a larger penalty). Hence, the MSE is selected instead of other loss functions such as mean absolute error (MAE) which provides equal error weights.

The selection of DNN models for MLP and LSTM can be summarised as follows First, grid search of hyper-parameters as shown in Fig. 7 is applied. The top 4 models with the smallest average errors (in terms of RMSE, R^2 and MAPE) are selected and the box plot is generated based on the selected models. The model with the highest consistency is selected (i.e., smallest median error and variance). The same model is considered for both LSTM and GRU as the two models are associated with time series prediction and we wanted to compare their accuracy.

VI. DATASET PREPROCESSING AND FEATURES EXTRACTION

This section describes the dataset preparations for supervised DNN as well as features extraction for improved prediction accuracy. Normalisation and scaling are usually applied to time series datasets before the training step so they have values between 0 and 1. This has the advantage of speeding up the convergence [48].

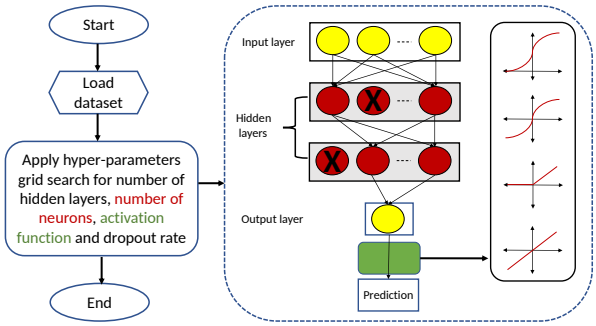


FIGURE 7. Grid search flowchart [28].

TABLE 1. Hyper-parameters settings for the grid search.

Hyper-parameter	Settings
Number of hidden layers	1, 2, 3, 4
Number of neurons	10, 30, 50, 80
Output activation function	Tanh, Linear, Sigmoid, ReLU
Dropout rate	1, 0.9, 0.75, 0.5
Batch size	128
Learning rate	0.001
Losses	MSE
Optimiser	Adam
Epochs	300

A. WALK FORWARD VALIDATION

The supervised training dataset is made using the 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 a real-life scenario where at every time a spectrum measurement is done and used to forecast the following DC.

B. FEATURES ENGINEERING

Features engineering is the process of extracting features from the raw dataset via pattern discovery [49]. In many cases, the estimated spectrum usage window needs to be longer than the original dataset. In a time series dataset, this means the original data set needs to be averaged and 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 proactive prediction significantly more complex. A solution based on including several input features besides the downsampled DC (Φ_c) is proposed to enable a proactive prediction.

In this work, we employ several statistical and

TABLE 2. Considered input features.

Feature name	Feature meaning
DC	DC values (Φ_c)
var	variance
slope	slope between the last two DC components (slope $= \Phi_K - \Phi_{K-1}$)
last DC value	last DC value Φ_K
3rd	skewness
4th	kurtosis
x_acf1	autocorrelation function of the series
diff1_acf1	autocorrelation function of the first-differenced series
diff2_acf1	autocorrelation function of the twice-differenced series
e_acf1	autocorrelation function of the residuals
x_acf10	sum of squares of the first 10 autocorrelation coefficients of the series
diff1_acf10	sum of squares of the first 10 autocorrelation coefficients for the first difference
diff2_acf10	sum of squares of the first 10 autocorrelation coefficients for the second difference
e_acf10	sum of squares of the first 10 autocorrelation coefficients for the residuals
entropy	Shannon entropy
crossing_points	number of times the temporal data-set crosses the median line
flat_spots	DC block is divided into ten equally blocks then the largest run length represent the value of flat_spots
nonlinear	nonlinearity coefficient estimated from a modified Teräsvirta's test
linearity	strength of linearity estimated from coefficients of the orthogonal quadratic regression
curvature	strength of curvature estimated from coefficients of the orthogonal quadratic regression
x_pacf5	sum of the first 5 partial autocorrelation coefficients of the series
diff1x_pacf5	sum of the first 5 partial autocorrelation coefficients first-order differenced series
diff2x_pacf5	sum of the first 5 partial autocorrelation coefficients second-order differenced series
lumpiness	variance of the means for non-overlapping windows
stability	variance of the variance for non-overlapping windows
arch_stat	statistic based on the Lagrange Multiplier test
trend	strength of trend estimated from Seasonal-Trend decomposition using LOESS
spike	variance of the leave-one-out variances of the residuals

information-theoretic measures recommended in [28], [29] and [50] to capture the relationship between past and future DC values (Φ_c). The engineered features are calculated for each block of duration T_c. For instance, the sample variance is found as from $Var(\Phi) = \frac{\sum_{k=1}^{K}(\Phi_k - \Phi_c)^2}{K}$. Other features include, slope between the last two DC components (slope$= \Phi_K - \Phi_{K-1}$), last DC value (i.e., Φ_K), skewness (3rd moment) and kurtosis (4th moment).

The autocorrelation function of the series (x_acf1), the first-differenced series (diff1_acf1), the twice-differenced series (diff2_acf1) and residuals autocorrelation (e_acf1). The sum of squares of the first 10 autocorrelation coefficients for series is x_acf10, for the first difference diff1_acf10, for the second order difference diff2_acf10 and residuals e_acf10. The spectral entropy is the Shannon entropy. The crossing_points is the number of times the temporal data-set crosses the median line. For flat_spots, the temporal data-set is divided into ten equally blocks then the largest run-length represents the value of flat_spots. The nonlinearity coefficient (nonlinear) is estimated from a modified Teräsvirta's test [51], it will have large values when the temporal data-set is nonlinear and small values when linear.

The strength of linearity and the strength of curvature are estimated from the coefficients of the orthogonal quadratic regression. The x_pacf5, diff1x_pacf5 and diff2x_pacf5, where pacf5 stands for the sum of the first 5 partial autocorrelation coefficients for the temporal data-set, the differenced series and the second-order differenced data-set, respectively. As for lumpiness and stability estimation, the temporal data-set is divided into tiled (non-overlapping) windows, the lumpiness is the variance of the mean of the tiled windows and stability is the variance of the variance of the tiled windows.

The arch_stat is the R^2 value of an autoregressive model of order specified as lags estimated from the Lagrange Multiplier test of Engle for autoregressive [52]. The measure of trend strength (trend) is found from Seasonal-Trend decomposition using LOESS [53]. Finally the variance of the leave-one-out variances of the remainder provides the strength of spikiness. Hence the input vector to the DNN will have the shape of $z \times a$, where z is the number of look back points and $a = 28$ is the number of input features. Table 2 summarises the included input features.

Even though [50] included several other statistical and information-theoretic features, from our experience we found them to have a minimal impact on the obtained results. Also, including not useful features will only impact the model training duration without improving the prediction accuracy. Thus, they will not be included. The computed input features are estimated for each DC block. More details on the considered features can be found in [28], [29] and [50].

C. MODEL INTERPRETATION

Explainable AI (XAI) is an emerging research field in machine and deep learning with the purpose of offering transparent interpretability for models. XAI main aim is to allow users to trust and manage successfully next-generation AI solutions [30].

One XAI approach is to utilise SHAP values to break down predictions to show the impact of each input feature on the output. SHAP is a unified approach to explain the output of any machine learning model. SHAP connects game theory with local explanations. The need to explain the output is important for DC prediction to limit the required parameters for accurate prediction and thus reducing model complexity. For the prediction problem, a variation of the original SHAP method called the DeepExplainer is optimised for explaining DNN models. More details on SHAP implementation can be found at [54].

VII. EXPERIMENTAL EVALUATION

A. MODELS EVALUATION

In this work, Python 3 language is used alongside Keras [55] API with TensorFlow [56] as backend. We tested the performance of the Adam [57] and Nadam [58] optimisers

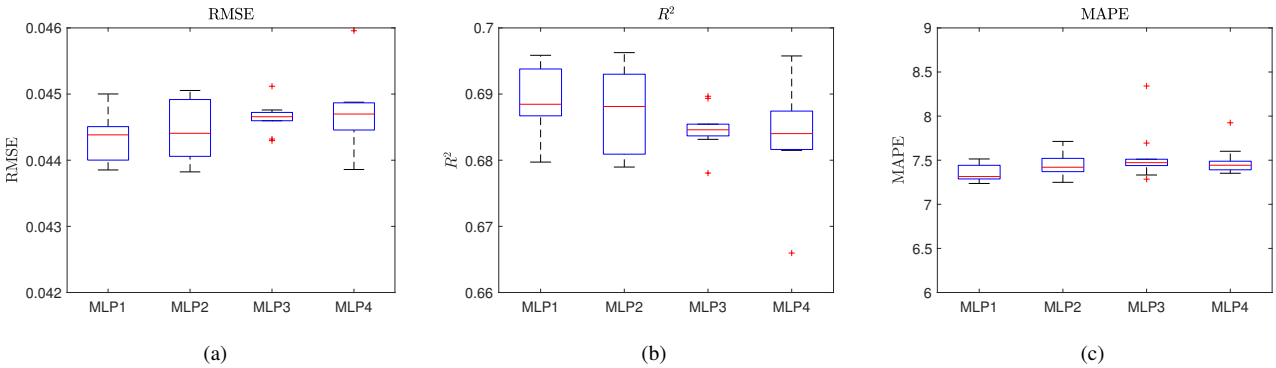


FIGURE 8. Accuracy for multiple MLPs (MLP1 4x50 tanh, MLP2 4x80 tanh, MLP3 4x30 tanh, MLP4 2x50 tanh), with $T_c = 30$ seconds and only DC as input feature. (a) RMSE, (b) R^2, (c) MAPE.

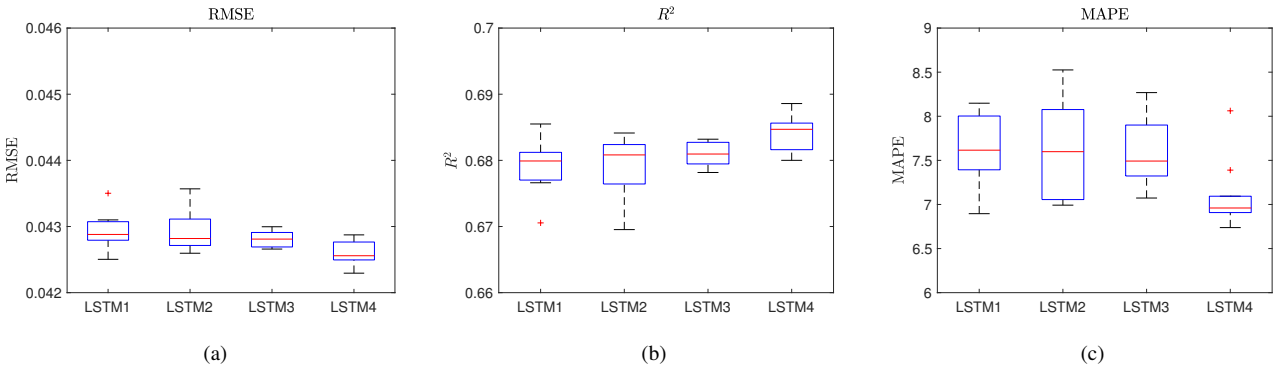


FIGURE 9. Accuracy for multiple LSTMs (LSTM1 = 4x50 sigmoid, LSTM2 4x80 sigmoid, LSTM3 3x80 sigmoid, LSTM4 4x80 tanh), with $T_c = 30$ seconds and only DC as input feature. (a) RMSE, (b) R^2, (c) MAPE.

and our results indicated that Adam tends to provide slightly better accuracy. Therefore, Adam is the optimiser that is used in this work. Noteworthy, Adam optimiser is by far the most commonly utilised optimiser in the current deep learning domain [8]. 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) with the same random seed values (from 1 to 10). The reported accuracy is averaged across the evaluations. A mini-batch size of 128 is used for the training and the learning rate = 0.001. All simulations were conducted with a maximum number of epochs set to 300. From the 13 measurement days, 6 days were used for training, 2 days for validation and 5 days for testing. An early stopping algorithm is utilised to prevent overfitting (during training stage) with an early stop value of 10 iterations [59]. At each prediction point, we firstly assume that 10 previous DC measurements are present when making the prediction (note that later a 15 measurements/lags will be considered as it provided better prediction accuracy). The results from ARIMA and DNNs are presented and compared in this section to predict the occupancy rate for different T_c values. The model performance was assessed by calculating RMSE, R^2 and MAPE metrics for the testing dataset.

First, we would like to investigate the impact of hyper-parameters optimisation on DNN performance. Figs. 8 and 9 show the prediction accuracy in terms of RMSE, R^2 and MAPE for MLP and LSTM, respectively. The best four models with the highest average accuracy are selected to be plotted in order to investigate which model performs better. T_c is set for 30 seconds and only DC input feature is considered at the input layer for DNN models. The middle line inside each box represents the median accuracy value and the lower and upper edges represent the first (Q_1) and third (Q_3) quartiles, respectively. The end of whiskers shown on the lower and upper sides are the minimum and maximum values, respectively. Accuracy values larger than $Q_3 + 1.5(Q_3 - Q1)$ or smaller than $Q_1 - 1.5(Q_3 - Q1)$ are considered as outliers [60]. Fig. 8 shows the performance of four MLP models with different architectures. The first considered MLP (MLP1) contains 4 hidden layers, 50 neurons in each layer and tanh activation function at the output layer. MLP2 contains 4 hidden layers, 80 neurons in each layer and tanh activation function at the output layer. MLP3 contains 4 hidden layers, 30 neurons in each layer and tanh activation function at the output layer. MLP4 contains 2 hidden layers, 50 neurons in each layer and tanh activation function at the output layer. MLP1 is found to provide the best outcome with no dropout for all of the considered metrics (i.e., RMSE,

TABLE 3. Best hyper-parameters for MLP and LSTM.

Hyper-parameter	MLP	LSTM
Number of hidden layers	4	4
Number of neurons	50	80
Hidden layers activation function	ReLU	Tanh
Output layer activation function	Tanh	Tanh
Dropout rate	1	1

R^2 and MAPE). As can be appreciated, MLP1 and MLP2 have similar median values, but with smaller variance and less complexity for MLP1. This observation is consistent for RMSE, R^2 and MAPE. Thus MLP1 configuration will be considered in the rest of this work and will be referred to as only MLP instead of MLP1.

The considered configurations for LSTM shown in Fig. 9 are as follows: LSTM1, LSTM2 and LSTM4 contain 4 hidden layers with 50, 80 and 80 neurons and sigmoid, sigmoid and tanh activation functions, respectively. LSTM3 contains 3 hidden layers with 80 neurons and sigmoid activation function. As can be concluded, LSTM4 provided the highest accuracy for the three metrics when compared to other LSTM models. Thus, LSTM4 configuration will be considered in the rest of this work and will be referred to as only LSTM instead of LSTM4. For GRU, a similar configuration is adopted to compare the performance with the LSTM architecture. Table 3 summarises the best hyper-parameters settings for MLP and LSTM.

Table 4 shows a comparison between different LSTM models against different T_c durations. 10 historical points (lags) are considered available at the model input with LSTM configuration as in LSTM4. Only DC stands for only having DC values at the input of the DNN. With features means beside the DC values, all statistical and information-theoretic are considered (explained in detail in Section VI-B). Taking first difference means, the input data have been differenced for prediction (i.e., the prediction will calculate the differenced DC $\Phi_{c,t} - \Phi_{c,t-1}$ value then correct it before estimating the accuracy metrics). Features and first difference means all input features and differenced DC values are considered.

As can be shown Table 4, for short-term DC prediction ($T_c < 3$ min), the proposed model of including first difference and several input features performs the best in terms of RMSE and R^2 when compared with other approaches. As for MAPE, considering only the first difference with no input features will have better predictions. While this result contradicts with RMSE and R^2 values, it suggests that including features and first difference minimises large errors as RMSE and R^2 provide a higher penalty for larger errors. Thus, including both input features and first difference are considered.

While it is suggested in the literature that only using features without taking difference is sufficient, Table 4 demonstrates otherwise. In order to obtain the best performance out of the LSTM model for dynamic access system, both statistical features and taking the first difference are required to improve the prediction accuracy. This observation holds for short DC prediction ($T_c < 3$ min).

Next, the effect of different lags (how many DC steps are available for each prediction) is investigated. Table 5 is considered for LSTM with first difference and all input features. When lags = 2, the model only has access to two previous DC values when trying to predict the future one. As expected, for this case, the accuracy is the worst for all three metrics. While having a large number of lags = 20, the prediction accuracy starts to degrade. The best performance is obtained when 15 lags are available at the input. Thus, it can be concluded that having 15 observations provides a reasonable trade-off between model complexity and accuracy and it will be considered in the rest of this work.

Next, we would like to compare the performance of proposed and normal deep neural networks with traditional ARIMA and SARIMA. The RW results are provided to serve as a baseline for obtained accuracy of different prediction methods. For ARIMA configuration, ARIMA(4,1,1) is found to provide the highest accuracy. While for SARIMA $(4,1,1)\times(1,1,1)_s$ provided the best results as explained in Section IV-B. Table 6, shows the prediction accuracy for all the considered methods in this work for different T_c values. ARIMA and SARIMA are generated using MATLAB [61]. SARIMA accuracy is only provided for $T_c \geq 1$ minutes as prediction with SARIMA requires a substantially long time when using large seasonality values. For instance, for prediction with $T_c = 1$ minute, the seasonality will be 1440 for SARIMA.

As can be appreciated, ARIMA and SARIMA have similar prediction accuracy with a slight advantage for ARIMA. This is as ARIMA is more suited for short prediction periods, short-term prediction is more random and high DC values occur mostly in short bursts, making seasonality relationship insignificant. SARIMA performs better for larger $T_c \geq 5$ minutes values which are out of the scope of this work.

MLP, GRU and LSTM stand for using DNN models with only DC available at the input. While GRU-proposed and LSTM-proposed have all input features (statistical and information-theoretic) at the DNN input. An interesting observation is that using GRU and LSTM alone does not provide any benefit over ARIMA. But adding several engineered input features improves the prediction accuracy for both LSTM and GRU by an average of 5% for RMSE metric when $T_c = 30$ seconds as an example. The proposed GRU and LSTM have similar performances with a slight advantage to LSTM in terms of accuracy for RMSE and R^2. But GRU generally has better accuracy in terms of MAPE. It can be concluded that the proposed feature-based LSTM and GRU architectures provide better performance for short-term prediction, as they provide an advantage when doing short-term prediction. This can be explained as the network traffic is correlated over short durations, therefore, having more features assist the model to improve prediction accuracy.

Figs. 10 and 11 show the forecasts of the considered mod-

TABLE 4. Comparison for LSTM and different input features.

Tc	Only DC			With features			Taking first difference			Features and first difference		
	RMSE	R^2	MAPE	RMSE	R^2	MAPE	RMSE	R^2	MAPE	RMSE	R^2	MAPE
10 sec	0.0495	0.6735	10.8596	0.0482	0.6913	11.1856	0.0484	0.6889	10.0974	0.047	0.7059	10.3037
30 sec	0.0443	0.6902	8.0297	0.0429	0.7085	8.3947	0.0443	0.6897	7.4761	0.0427	0.711	7.6748
45 sec	0.0435	0.6839	7.5359	0.0432	0.6872	8.9731	0.0427	0.6956	6.9215	0.0427	0.6951	7.5148
1 min	0.0426	0.6843	7.0857	0.042	0.6936	8.2786	0.0424	0.6863	6.6436	0.0418	0.6952	7.035
3 min	0.0409	0.6614	8.507	0.0441	0.6047	9.8404	0.0396	0.6816	6.3397	0.0436	0.6134	6.9284
5 min	0.0439	0.5909	9.2139	0.0478	0.5135	10.4926	0.0437	0.5938	6.6057	0.0447	0.5745	6.9153

TABLE 5. Comparison for LSTM and different lags size.

Tc	Lags = 2			Lags = 5			Lags = 10			Lags = 15			Lags = 20		
	RMSE	R^2	MAPE	R^2	R^2	MAPE	RMSE	R^2	MAPE	RMSE	R^2	MAPE	RMSE	R^2	MAPE
10 sec	0.0481	0.6917	11.0469	0.0472	0.7039	10.6584	0.047	0.7059	10.3037	0.0467	0.7099	9.9322	0.0470	0.7066	10.0718
30 sec	0.0433	0.7031	8.0076	0.0428	0.7099	7.8718	0.0427	0.711	7.6748	0.0427	0.7119	7.6921	0.0426	0.7132	7.6828
45 sec	0.0429	0.6917	7.6965	0.0429	0.6925	7.4674	0.0427	0.6951	7.5148	0.0424	0.6989	7.5813	0.0424	0.6995	7.4334
1 min	0.0419	0.6939	6.9749	0.0419	0.6943	6.9878	0.0418	0.6952	7.035	0.0417	0.6977	6.9241	0.0414	0.7018	6.8581
3 min	0.0404	0.669	6.8555	0.0433	0.62	6.9405	0.0436	0.6134	6.9284	0.0431	0.6233	6.728	0.0425	0.6324	6.6716
5 min	0.0442	0.5839	7.0492	0.0439	0.5892	6.9424	0.0447	0.5745	6.9153	0.0441	0.5858	7.0319	0.0448	0.5722	6.8067

TABLE 6. The achieved performance metrics, Table 6(a), 6(b) and 6(c) shows the RMSE, R^2 and MAPE metrics, respectively.

(a)

Tc	RW	ARIMA	SARIMA	MLP	GRU	GRU-proposed	LSTM	LSTM-proposed
10 sec	0.056	0.0494	-	0.0487	0.0487	0.047	0.0497	0.0467
30 sec	0.049	0.0444	-	0.0446	0.0447	0.0427	0.0444	0.0427
45 sec	0.0477	0.0433	-	0.0435	0.0434	0.0424	0.0437	0.0424
1 min	0.0467	0.0426	0.0459	0.0429	0.0426	0.0418	0.0425	0.0417
3 min	0.0417	0.0397	0.0407	0.0416	0.0401	0.0414	0.0409	0.0431
5 min	0.0452	0.043	0.0428	0.0465	0.0429	0.0437	0.0439	0.0441

(b)

Tc	RW	ARIMA	SARIMA	MLP	GRU	GRU-proposed	LSTM	LSTM-proposed
10 sec	0.583	0.6747	-	0.6841	0.6848	0.7065	0.6718	0.7099
30 sec	0.6202	0.6887	-	0.6854	0.6843	0.7121	0.6879	0.7119
45 sec	0.6197	0.6867	-	0.6834	0.685	0.6997	0.6808	0.6989
1 min	0.6206	0.6837	0.6339	0.6795	0.684	0.6962	0.686	0.6977
3 min	0.6469	0.6798	0.6638	0.6488	0.6745	0.6525	0.6609	0.6233
5 min	0.565	0.6062	0.6107	0.5407	0.6092	0.5933	0.5909	0.5858

(c)

Tc	RW	ARIMA	SARIMA	MLP	GRU	GRU-proposed	LSTM	LSTM-proposed
10 sec	12.4811	10.5513	-	10.4968	10.4083	10.2789	11.0329	9.9322
30 sec	8.6236	7.7179	-	7.5922	8.2297	7.6437	8.1885	7.6921
45 sec	7.9966	7.2106	-	7.105	7.5873	7.3807	7.7782	7.5813
1 min	7.3298	6.8316	8.7591	6.8043	7.2452	6.8934	6.9997	6.9241
3 min	6.6103	6.3856	7.926	6.9243	7.661	6.8269	8.3832	6.728
5 min	6.6089	6.5212	8.2701	7.643	7.7305	7.2669	8.6437	7.0319

els over $T_c = 30$ seconds for different time indices. As can be seen, the proposed feature-based models (for both LSTM and GRU) show better ability to adapt to sharp fluctuations in the actual DC (measured) than ARIMA, MLP, original GRU and LSTM. This makes the proposed model suitable for DC predictions with high fluctuations.

Fig. 12 shows the forecasts of the considered models over $T_c = 3$ minutes. The measured DC shows a small volatility trend. Nevertheless, the proposed LSTM and GRU methods still perform well.

B. EXPLAINABLE ARTIFICIAL INTELLIGENCE

In order to get an overview of the impact of various input features for the model, we have plotted the SHAP values for all input features used at the DNN input. Fig. 13 shows the absolute feature importance for all steps, the y-axis is the importance of each feature and their total sum is 1. The input features are sorted from left to right in a descending order of importance. The most important feature is past DC values to determine future DC values (i.e., $(\Phi_{c,t-1}...\Phi_{c,t-L})$ to decide $\Phi_{c,t}$, where L is number of lags), followed by (diff1_acf1). Fig. 14, shows the SHAP values sorted according to importance (from top to bottom) for the case of first step only

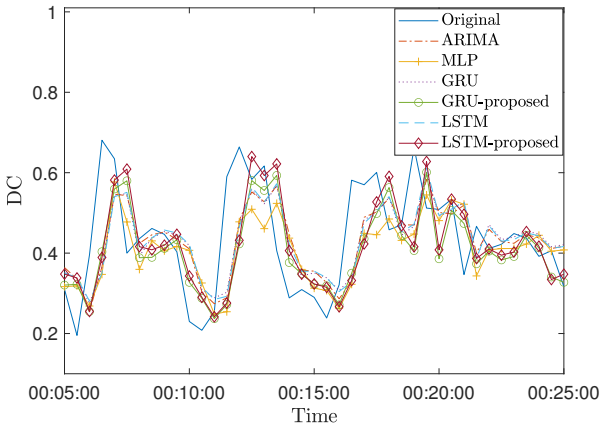


FIGURE 10. Prediction for $T_c = 30$ seconds.

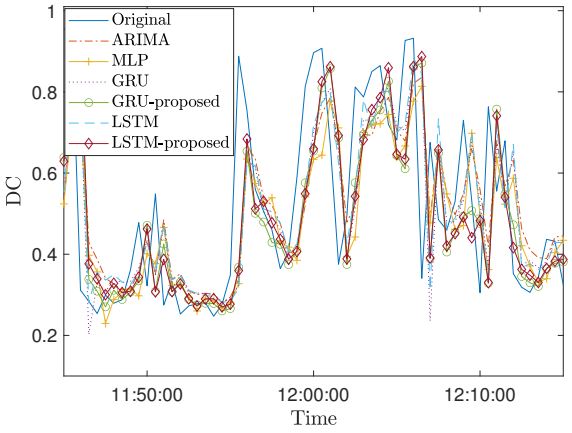


FIGURE 11. Prediction for $T_c = 30$ seconds.

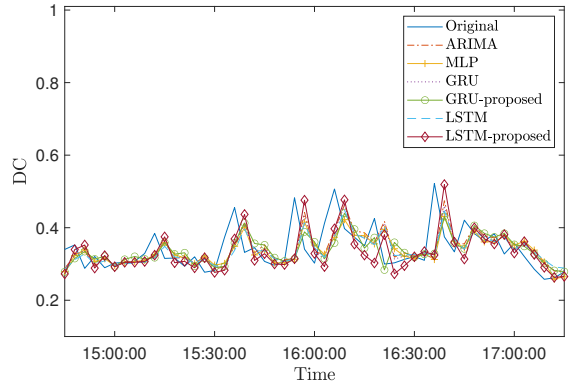


FIGURE 12. Prediction for $T_c = 3$ minutes.

(first lag). Only 12 input features are plotted as the rest of the features have a negligible contribution. Fig. 14 shows the distribution of impacts for every feature on the model output. The colour represents the feature value (red means high, blue means low). This reveals for example, high historical DC values (shown in red) tend to increase the predicted DC value. Fig. 14 also implies that larger linearity, variance and last value tend to increase the predicted DC value of the DNN model. While smaller linearity, variance and last value tend to decrease the predicted DC value.

Fig. 15 shows the absolute feature importance with the first difference DC and taking the effect of all lags. The most important feature is also past DC values to determine future DC values. But this time, it has 20% importance instead of 35%. The decrease in DC importance is shared by other features, by having their importance slightly increased. Moreover, linearity and last value (Φ_k) are the second and third features in importance, respectively. While in Fig. 13, diff1_acf1 and var features are the second and third features in importance. As it can be concluded from Figs. 13 and 15, taking the first difference changes the relationship between historical and future DC values, thus features importance would vary based on the considered preprocessing approach.

Fig. 16 shows the SHAP values sorted according to importance for the case of taking first difference DC for the first step only and for 12 input features. Fig. 16 implies that high historical DC values (shown in red) tend to result in a decrease of the predicted DC. This is explained as the model tries to predict the differenced DC ($\Phi_{c,t} - \Phi_{c,t-1}$). Thus, it has a negative effect on the predicted values. Fig. 16 also implies that large linearity values increase the predicted differenced DC, while large values of x_acf1 reduce the predicted differenced DC values.

Based on features with importance of 5% or larger from Fig. 15, we use them as input features (5 features namely DC, linearity, last_val, diff1x_pacf5 and x_acf1) for the LSTM model to predict the differenced DC. This approach provides the benefits of reducing the DNN complexity from using all input features.

The complexity of a DNN can be measured by different approaches. One approach is based on the number of trainable parameters in a DNN [62]. Here, the complexity is measured as a function of trainable parameters, since the algorithm runtime scales linearly with it. Thus, less number of parameters means a less complex DNN. The second proposed method (LSTM-proposed 2) reduces the number of trainable parameters by 4% from when all input features are used (LSTM-proposed) as can be seen in Table 7.

Another important complexity parameter is the time complexity of a DNN. In general, time complexity can be divided into training (offline) and prediction (online) times. The prediction time also includes any preprocessing or data cleaning. For dynamic spectrum access systems, prediction time is more essential as once the DNN model is trained, it can be used directly with no changes to the architecture. In practice, new DC measurements will be available from

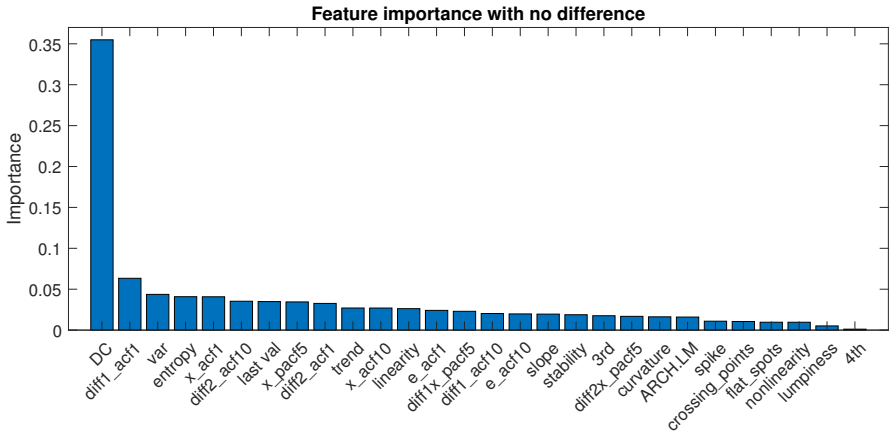


FIGURE 13. Relative importance of each feature used for prediction in case of no difference is considered for LSTM model with 15 lags and $T_c = 30$ seconds.

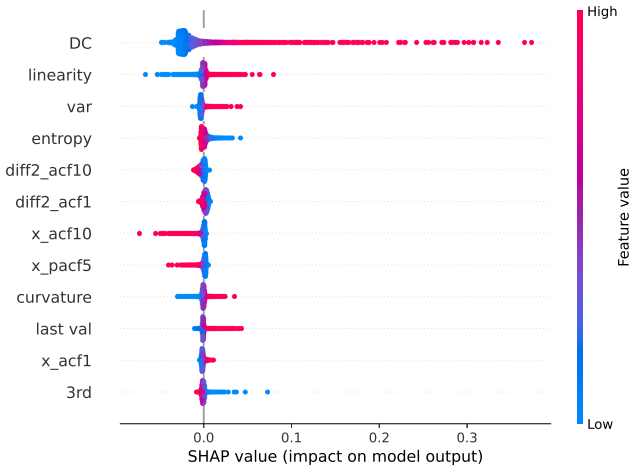


FIGURE 14. Relative importance of each feature used for prediction in case of no difference is considered.

sensors. The first step is to do preprocessing and features extraction. Shorter preparation time means the faster the data will be available at the DNN input. For the case of considering all 28 input features, the preprocessing time is an average of 9.3 seconds. While when only considering the top 5 features, the required time is 2.98 seconds. Hence a shorter time for data preparation. This indicates the importance of understanding each feature's contribution to the prediction outcome to select only useful features. The execution time for this part was conducted using Google Colaboratory [63] and [64].

Lastly, Fig. 17 shows the performance of the second proposed model based on 5 input features (LSTM-proposed 2) versus normal LSTM (Only DC at input) and LSTM with all input features (LSTM-proposed). The second proposed model shows the ability to adapt to high fluctuations in a similar manner to having all input features but with less complexity.

TABLE 7. Proposed models complexity and accuracy.

LSTM Model	LSTM (DC only)	LSTM-proposed (All features)	LSTM-proposed 2 (5 features)
Trainable parameters	182,161	190,801	183,441
RMSE	0.0444	0.0427	0.0423
R^2	0.6879	0.7119	0.7170
MAPE	8.1885	7.6921	7.4361

VIII. CONCLUSION

A proactive spectrum usage estimator is essential for flexible next generation-systems. In this work, a feature-based DNN for short-term DC prediction is proposed and investigated. Several prediction durations for DC are investigated and analysed using several performance assessment metrics to have a full understanding for the performance of the considered prediction methods. First, we studied the prediction accuracy of several DNN models (MLP, LSTM and GRU) and showed that the MLP model had the worst performance among the considered DNN models and LSTM/GRU are more suited for time series data. Then, we showed that using deep learning algorithms directly (only DC at the input for the DNN) does not provide noticeable prediction improvement with respect to ARIMA/SARIMA models. Moreover, only considering the first difference still does not provide significant improvement to the prediction accuracy over traditional temporal prediction algorithms such as ARIMA. The best performance from LSTM/GRU is obtained when both the first difference and engineered input features are considered. Thus, several engineered input features are considered at the input of LSTM and GRU to improve the prediction accuracy. Moreover, to increase the trust in the obtained results from DNN, SHAP values are used to demonstrate the contribution of each input feature on the resulting DNN output (i.e., DC). While previous occupancy rate has the highest weight in deciding future values of occupancy rate, including other input features also contribute to further improving the prediction accuracy and makes the DNN model more versatile in handling large variations in DC values. Finally, based on the level of importance, a simplified model was shown to provide

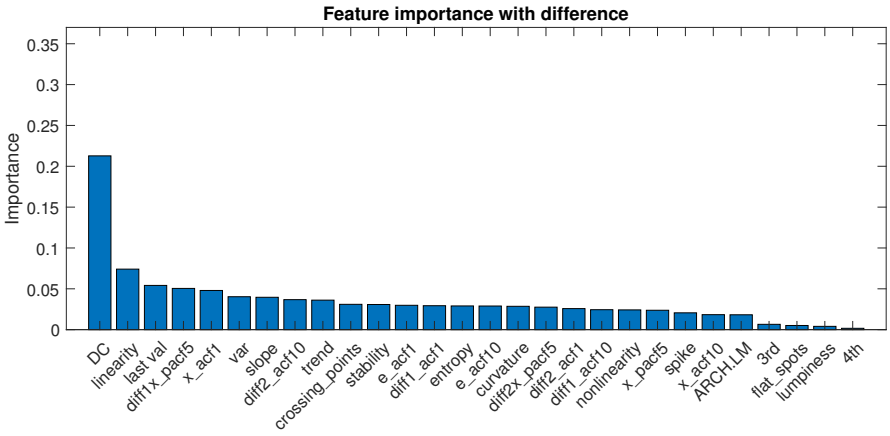


FIGURE 15. Relative importance of each feature used for prediction in case of first order difference is considered for LSTM model with 15 lags and $T_c = 30$ seconds.

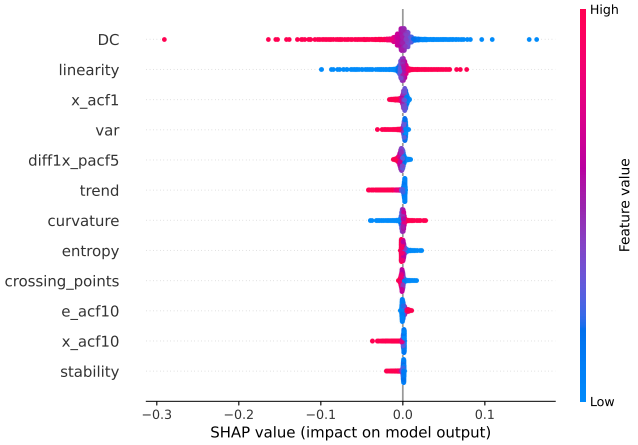


FIGURE 16. Relative importance of each feature used for prediction in case of first order difference is considered.

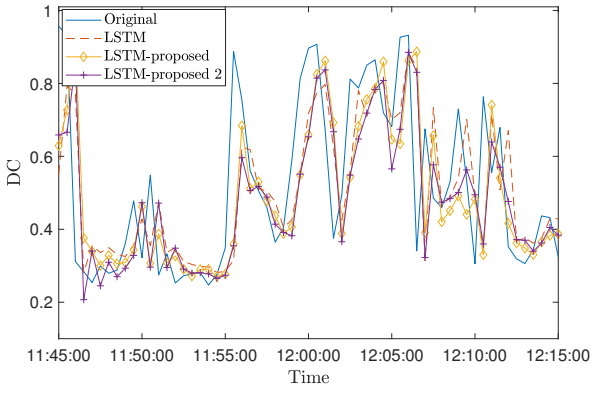


FIGURE 17. Prediction for $T_c = 30$ seconds.

comparable accuracy to using all input features. Future work will include the investigation of other deep learning models such as Bi-LSTM and Bi-GRU with multiple input features and the analysis of their potential capability to improve the DC prediction accuracy.

REFERENCES

[1] "Mobile data traffic outlook," Ericsson, Tech. Rep., July 2020. [Online]. Available: https://www.ericsson.com/en/mobility-report/reports/june-2020/mobile-data-traffic-outlook
[2] T. Fujii and K. Umebayashi, "Smart spectrum for future wireless world," *IEICE Transactions on Communications*, vol. E100B, no. 9, pp. 1661–1673, Sep. 2017.
[3] Y. Zhou, Z. M. Fadlullah, B. Mao, and N. Kato, "A deep-learning-based radio resource assignment technique for 5G ultra dense networks," *IEEE Network*, vol. 32, no. 6, pp. 28–34, 2018.
[4] 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.
[5] Y. Shu, Z. Jin, L. Zhang, L. Wang, and O. Yang, "Traffic prediction using FARIMA models," in *1999 IEEE International Conference on Communications*, vol. 2, 1999, pp. 891–895.
[6] S. Medhn, B. Seifu, A. Salem, and D. Hailemariam, "Mobile data traffic forecasting in UMTS networks based on sarima model: The case of addis ababa, ethiopia," in *2017 IEEE AFRICON*, 2017, pp. 285–290.
[7] C. Zhang, H. Zhang, J. Qiao, D. Yuan, and M. Zhang, "Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1389–1401, 2019.
[8] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.
[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] J. E. Smee and J. Hou, "5G+AI : The ingredients fueling tomorrow's tech innovations," Qualcomm, Tech. Rep., 02 2020. [Online]. Available: http://www.qualcomm.com/news/onq/2020/02/04/5gai-ingredients-fueling-tomorrows-tech-innovations
[13] 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.
[14] 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.
[15] "White Paper: 5G Evolution and 6G," NTT DOCOMO, Tech. Rep., 01 2020.

[16] 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.

[17] T. H. H. Aldhyani, M. Alrasheedi, A. A. Alqarni, M. Y. Alzahrani, and A. M. Bamhdi, "Intelligent hybrid model to enhance time series models for predicting network traffic," *IEEE Access*, vol. 8, pp. 130 431–130 451, 2020.

[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[19] A. Lazaris and V. K. Prasanna, "Deep learning models for aggregated network traffic prediction," in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[20] Q. Duan, X. Wei, Y. Gao, and F. Zhou, "Base station traffic prediction based on STL-LSTM networks," in *2018 24th Asia-Pacific Conference on Communications (APCC)*, 2018, pp. 407–412.

[21] S. Sevgican, M. Turan, K. Gökarslan, H. B. Yilmaz, and T. Tugcu, "Intelligent network data analytics function in 5G cellular networks using machine learning," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 269–280, 2020.

[22] 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.

[23] 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.

[24] L. Yu, Q. Wang, Y. Guo, and P. 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.

[25] S. P. Sone, J. J. Lehtomäki, and Z. Khan, "Wireless traffic usage forecasting using real enterprise network data: Analysis and methods," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 777–797, 2020.

[26] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," ser. Mobihoc '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 231–240.

[27] B. Pfülb, C. Hardegen, A. Gepperth, and S. Rieger, "A study of deep learning for network traffic data forecasting," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham: Springer International Publishing, 2019, pp. 497–512.

[28] A. Al-Tahmeesschi, K. Umebayashi, H. Iwata, M. López-Benítez, and J. Lehtomäki, "Applying deep neural networks for duty cycle estimation," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–7.

[29] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015, pp. 1616–1619.

[30] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, no. 2, 2017.

[31] G. Vilone and L. Longo, "Explainable artificial intelligence: A systematic review," *arXiv:2006.00093*, 2020.

[32] Z. Quan-shi and Z. Song-chun, "Visual interpretability for deep learning: A survey," *Frontiers of Information Technology and Electronic Engineering*, vol. 19, pp. 27–39, 2018.

[33] J. X. Mi, A. D. Li, and L. F. Zhou, "Review study of interpretation methods for future interpretable machine learning," *IEEE Access*, vol. 8, pp. 191 969–191 985, 2020.

[34] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[35] G. Fidel, R. Bitton, and A. Shabtai, "When explainability meets adversarial learning: Detecting adversarial examples using shap signatures," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[36] A. Ghasemi, "Predictive modeling of LTE user throughput via crowdsourced mobile spectrum data," in *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2018, pp. 1–5.

[37] R. Nau, "Introduction to forecasting, the simplest models : Notes on the random walk model," *Duke University*, 2014. [Online]. Available: https://people.duke.edu/~rnau/Notes_on_the_random_walk_model--Robert_Nau.pdf

[38] G. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[39] R. Nau, "Arima models for time series forecasting: Slides on seasonal and nonseasonal arima models," *Duke University*, 2014. [Online]. Available: https://people.duke.edu/~rnau/Slides_on_ARIMA_models--Robert_Nau.pdf

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

[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

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

[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014.

[44] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, pp. 679–688, 2006.

[45] 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.

[46] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, Jun 2016.

[47] J. Ruiz-Aguilar, I. Turias, and M. Jiménez-Come, "Hybrid approaches based on sarima and artificial neural networks for inspection time series forecasting," *Transportation Research Part E: Logistics and Transportation Review*, vol. 67, pp. 1 – 13, 2014.

[48] E. P. Lens Shiang, W. Chien, C. Lai, and H. Chao, "Gated recurrent unit network-based cellular trafile prediction," in *2020 International Conference on Information Networking (ICOIN)*, 2020, pp. 471–476.

[49] B. Boehmke and B. M. Greenwell, "Hands-on machine learning with R." CRC Press, 2019.

[50] Rob Hyndman, Yanfei Kang, Pablo Montero-Manso, Thiyanga Talagala, Earo Wang, Yangzhuoran Yang, Mitchell O'Hara-Wild, Souhaib Ben Taieb, Cao Hanqing D K Lake, Nikolay Laptev, J R Moorman, *Time Series Feature Extraction*, R Foundation for Statistical Computing, 2020. [Online]. Available: https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html#license

[51] T. Teräsvirta, C.-F. Lin, and C. W. J. Granger, "Power of the neural network linearity test," *Journal of Time Series Analysis*, vol. 14, no. 2, pp. 209–220, 1993. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1993.tb00139.x

[52] P. Catani and N. Ahlgren, "Combined lagrange multiplier test for arch in vector autoregressive models," *Econometrics and Statistics*, vol. 1, pp. 62–84, 2017.

[53] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[54] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim *et al.*, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomedical Engineering*, vol. 2, no. 10, p. 749, 2018.

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

[56] 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.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[58] T. Dozat, "Incorporating nesterov momentum into Adam," in *International Conference on Learning Representations (ICLR)*, 2016.

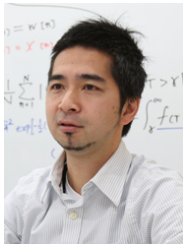
[59] L. Prechelt, *Early Stopping — But When?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_5

[60] R. R. Wilcox, *Applying Contemporary Statistical Techniques*. Academic Press, 2003.

[61] The MathWorks Inc., *Econometric Modeler*, Natick, Massachusetts, United State, 2020. [Online]. Available: https://www.mathworks.com/help/econ/econometric-modeler-overview.html

[62] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.
[63] Google, "Colaboratory: Frequently asked questions," 2021. [Online]. Available: https://research.google.com/colaboratory/faq.html
[64] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance analysis of google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61 677–61 685, 2018.

AHMED AL-TAHMEESSCHI received the B.Sc. and M.Sc. degrees in communications engineering from the University of Technology, Baghdad, Iraq, in 2009 and 2011, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Electronics, University of Liverpool, United Kingdom in 2018. Currently, he is a postdoctoral researcher at the Tokyo University of Agriculture and Technology. His research interests include cognitive radio networks, dynamic spectrum access techniques, algorithm design, and machine learning.



KENTA UMEBAYASHI received the B.E., M.E., and Ph.D. degrees from Yokohama National University, Japan, in 1999, 2001, and2004, respectively, and the L.L.B. degree from Ritsumeikan University, Japan, in 1996. From 2004 to 2006, he was a Research Scientist with the Centre for Wireless Communications, University of Oulu, Finland. He is currently a Professor with the Tokyo University of Agriculture and Technology, Japan. He was a Principal Investigator of four grants-in-aid for scientific research projects and three strategic information and communications research and development promotion programme projects, including a HORIZON2020 project. His research interests lie in the areas of signal detection and estimation theories for wireless communication, signal processing for multiple antenna systems, cognitive radio networks, and Terahertz band wireless communications. He received the Best Paper Award at 2012 IEEE WCNC and the Best Paper Award at 2015 IEEE WCNC Workshop from IWSS.

HIROKI IWATA was born in Miyazaki, Japan in 1992. He received his B.E., M.E. and Ph.D. degrees in electrical and electronic engineering from the Tokyo University of Agriculture and Technology, Tokyo, Japan, in 2014, 2016 and 2019, respectively. Currently, he is a postdoctoral researcher at the Tokyo University of Agriculture and Technology. His research interests are cognitive radio networks, spectrum measurement and wireless communication systems. He received the Best Paper Award in International Workshop on Smart Spectrum (IWSS) at IEEE WCNC 2016 for a paper he authored.

JANNE LEHTOMÄKI got his doctorate from the University of Oulu, Oulu, Finland in 2005. Currently, he is an adjunct professor (docent) at the University of Oulu, Centre for Wireless Communications. He spent the fall 2013 semester at Georgia Tech, Atlanta, as a visiting scholar. Currently, he is focusing on cognitive radios and terahertz band wireless communication. He co-authored the paper receiving the Best Paper Award at IEEE WCNC 2012. He is an Associate Editor of Physical Communication.

MIGUEL LÓPEZ-BENÍTEZ received the BSc and MSc degrees (both with Distinction) in Telecommunication Engineering from Miguel Hernández University, Elche, Spain in 2003 and 2006, respectively, and the PhD degree (*summa cum laude*) in Telecommunication Engineering from the Technical University of Catalonia, Barcelona, Spain in 2011. From 2011 to 2013, he was a Research Fellow with the Centre for Communication Systems Research, University of Surrey, Guildford, UK. In 2013, he became a Lecturer (Assistant Professor) with the Department of Electrical Engineering and Electronics, University of Liverpool, UK, where he has been a Senior Lecturer (Associate Professor) since 2018. His research interests include the field of wireless communications and networking, with special emphasis on mobile communications, dynamic spectrum access and the Internet of Things. Personal website at http://www.lopezbenitez.es.