Neural Network Based Spectrum Prediction in Land Mobile Radio Bands for IoT deployments

Sanaz Mohammadjafari, Emir Kavurmacioglu, John Maidens, Ayse Bener
Data Science Laboratory
Ryerson University, Toronto, Canada
{sanaz.mohammadjafari,emir,maidens,ayse.bener}@ryerson.ca

***Abstract*—Aim: We seek to assess the performance of time delay neural networks (TDNN), one of the topologies designed for time series prediction, to characterize spectrum occupancy in multiple time horizons in Land Mobile Radio bands. This could lead to dynamic spectrum allocation methods to address potential spectrum shortages facing Internet of Things (IoT) deployments. Background: ANNs are a popular choice for spectrum prediction. Traditionally, ARIMA models have been at the forefront of forecasting and prediction but ANNs that learn from time series have demonstrated good performance using both simulated datasets and real-life data collected in the cellular bands. Methodology: We use three prediction models, a baseline which simply delays the time series, a seasonal ARIMA model and a TDNN. We test their performance on an hourly dataset in LMR bands collected in Ottawa, Canada between the dates of October 2016 and April 2017. Results: We demonstrate that TDNN yields improvements over seasonal ARIMA models in predicting short time horizons. Conclusions: The TDNN based prediction models that are designed to work with time series data provide a better alternative for accurately predicting spectrum occupancy in bands that exhibit similar characteristics to LMR channels, especially as the forecast horizon gets longer.**

***Index Terms*—Dynamic Spectrum Prediction, Artificial Neural Networks, ARIMA, Time Delay Neural Networks, Internet of Things**

I. Introduction

Technologies such as Internet of Things (IoT) increase the demand for communication spectrum which is a scarce resource. The current spectrum allocation policies may be the reason for this bandwidth scarcity due to inefficient use of available radio spectrum bands. Dynamic Spectrum Access (DSA), a multi-tier hierarchical ownership scheme [1], is one of the methods proposed to overcome this problem.

Land Mobile Radio (LMR) band that range from 138 to 941 MHz, would potentially be a good candidate for implementing DSA. Measurement studies conducted in Chicago indicate that LMR channels are utilized with varying occupancy rates. This implies a potential for more efficient use of the spectrum [2]. The FCC primarily allocates LMR channels for voice communications through commercial and federal or non-federal government agencies/services [3]. LMR spectrum bands are used for servicing first responder organizations such as police, fire, and ambulance services, public works organizations like utility companies, dispatched services such as taxis, or other companies with large vehicle fleets.

978-3-903176-15-7

Moving even small bands of spectrum to a dynamic access regime cannot be done manually. There is a need for an intelligent and learning based system such as artificial intelligence (AI) techniques to support real time prediction of spectrum usage. Cognitive Radios (CR) appear to be a potential solution to solve the problem of inefficient spectrum usage and spectrum scarcity among unlicensed users [4]–[8]. In a CR environment, licensed users are called primary users (PU) and the unlicensed users are called secondary users (SU). In CR, in an effort to minimize the interference between an incumbent primary user, a potential secondary user would be required to sense the entire spectrum band to detect those channels in the spectrum which might be idle. Since the sensing process is altogether time and energy consuming, it creates an unnecessary burden in the IoT environment, where most devices are operating under a battery constraint. A reliable prediction regime that indicates those channels which are going to be suitable for transmission based on the previous sensing history would address this problem.

Various analytical models have been proposed to predict primary user behavior using temporal data, including linear prediction models such as auto regressive integrated moving average (ARIMA) [9], [10] that are rooted in the domain of traditional time series analysis and artificial neural networks (ANN) and its variants from the domains of machine learning and artificial intelligence [11]–[13]. In this paper we seek to understand whether the addition of a non-linear component to time series prediction provides any measurable benefits over linear prediction methods. In this paper we show an improvement in prediction performance in short time horizons when using time delay neural networks (TDNN), a non-linear model, over seasonal ARIMA, which are linear models.

The remainder of the paper is structured as following: In Section II we provide an overview of various models used in spectrum prediction. In Section III we provide the dataset used in this paper as well as the pre-processing and exploratory analysis performed on the dataset. Also, in this section, we provide the prediction models used in the paper, namely the seasonal ARIMA and the TDNNs and the methodology followed in their training and calibration. In Section IV, we present the results and discuss the findings and presents some future direction for the paper in Section V.

II. Related Work

In the literature, there are many proposed models and approaches to predict the occupancy status in a channel. Ding et. al. provide a thorough survey of models that are developed for spectrum inferences in various dimensions [11].

Some of the most widely studied and oldest prediction models are linear prediction models, where future values are predicted as linear function of past observations. Due to their simplicity they are very popular and have been widely used in various fields, including spectrum prediction [9], [10], [14]–[17]. Most common models include auto-regressive (AR), moving average (MA), and auto-regressive integrated moving average (ARIMA) models.

Another popular prediction family of methods are ANNs, which represent a class of flexible non-linear models [12], [17]–[21]. Yin et. al. proposed using a TDNN to predict the max-min normalized power measurements of channels [20] and report an improved root-mean-square error (RMSE) compared to using ARIMA. The authors emphasize the nonlinear nature of the real-life spectrum data and the difficulty of modeling this behavior using mathematical models such as a Poisson process. The authors have collected non-channelized measurements ranging between 20MHz - 3GHz over a week and divided them into channels with a bandwidth of 100KHz. However, Stolojescu-Crisan in [17] shows that while NN based predictions outperform ARIMA when future horizon is relatively small, ARIMA is able to learn medium and long term temporal behaviors better.

The majority of the research encountered in the literature use simulated data which might not reflect an objective evaluation of the performance of prediction models. Real life studies on the other hand concentrate on cellular traffic which follows daily cyclical patterns and thus differs from spectrum activity in other parts of the radio spectrum. There is a scarcity of real life measurements and the evaluation of the performance of spectrum prediction models in the LMR spectrum.

III. Methodology

A. Data

The data used in this paper consists of a measurements made on the 8,663 LMR channels within LMR spectrum bands in downtown Ottawa between October 27th 2016 to April 26th 2017. LMR bands lie in the 138-174 MHz (VHF), 406-470 MHz (UHF) and in the 800 MHz (exact range purposefully omitted) ranges. The bandwidth of channels range between 4-280 KHz, with the majority of the channels having a bandwidth less than 10 KHz.

For each channel, a set of measurements on power (dBm), SNR (dB), bandwidth (Hz) and carrier frequency (Hz) are taken and stored every 300 milliseconds. If the power level does not exceed a certain dynamically adjusted threshold value on that specific channel, the sensors omit that measurement. In a given hour, if a channel is used nonstop then it will have approximately 12,000 observations. However, due to omitted measurements, a channel can have fewer observations. This

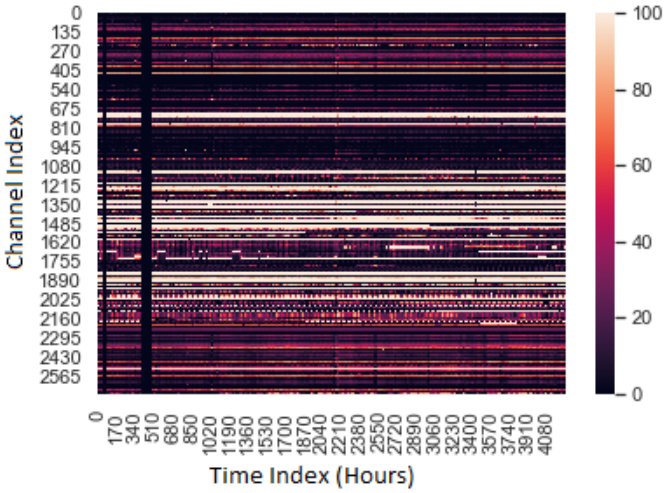

Fig. 1: The heat-map shows the occupancy % values of the channels which are more than %95 complete. Vertical dark lines indicate the time ranges where no data is recorded for any of the channels due to hardware issues

yields the hourly *occupancy percentage (%)* of a channel and is calculated by

$$\text{occupancy } \% = \frac{\#\text{ of recorded observations}}{\#\text{ of expected observations}} \quad (1)$$

The heat-map of channels occupancy in the given time range provided in Fig. 1 indicates that not all channels had a contiguous set of occupancy %s.To address this issue, we used the channels that report occupancy measurements most of the time (+95%). This leaves 2684 channels[1]. This threshold was chosen to leave us a sizable channels list (approximately 31% of all channels) while removing most of the incomplete observations from the data set.

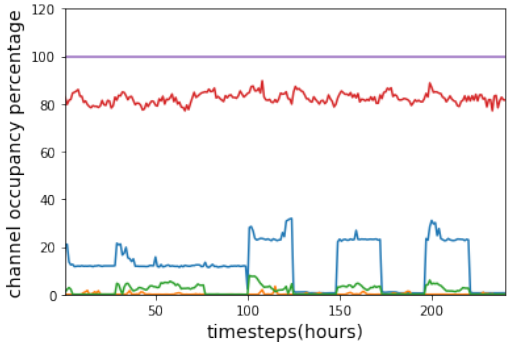

Fig. 2: The occupancy % of five random channels

Fig. 2 illustrates the temporal occupancy behavior of 5 randomly selected channels out of the selected 70 channels over 240 hours. An initial look indicates periodic behavior in some of the channels, e.g., channel 1. To confirm the existence of periodic patterns, we calculated the auto-correlation function (ACF) of these 5 channels that is provided in Fig. 3. The

[1]Note that this is different than channels with 0% occupancy, which would be reported in the measurement data.

peaks are confirming the periodic pattern in channel occupancy %. To train the models, we normalized the occupancy %s to values between $[0, 1]$.

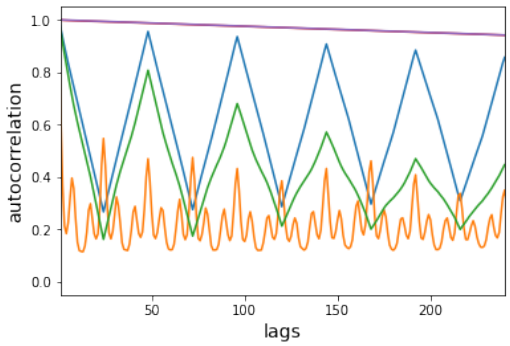

Fig. 3: The ACF of five random channels

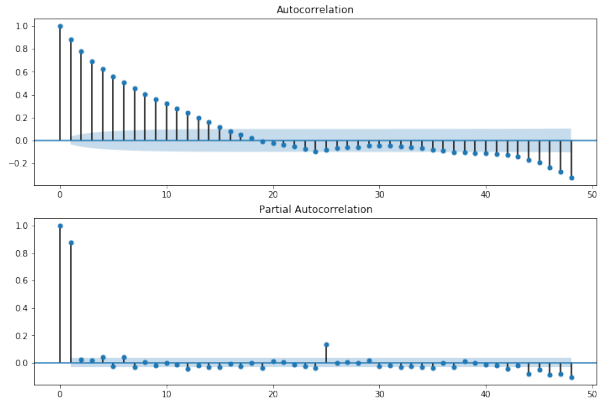

Fig. 4: The ACF and PACF of a channel in our dataset, which exhibits a highly characteristic AR signature with 2 autoregressive terms

B. Model

1) ARIMA: One of the most popular linear models for time series prediction is the auto-regressive integrated moving average (ARIMA) model [11]. The general ARIMA model (p, d, q) is formulated as below:

$$W(t) = \sum_{i=1}^{p} \alpha_i W(t-i) + \sum_{i=0}^{q} \beta_i e(t-i),$$

where $W(t) = X(t) - X(t-d)$, α_i and β_i are the linear coefficients of the model, $e(t)$ represents the errors with respect to the mean, and d is the degree of non-stationary homogeneity.

The seasonal part of an ARIMA model follows the same structure as the non-seasonal part $(P, D, Q)_s$,it may have an AR factor P, an MA factor Q, and an order of differentiating, D. In the seasonal part of the model, all of these factors operate across multiples of a seasonality parameter s, which indicates the number of time steps before starting a new seasonal cycle. Seasonal part of ARIMA model $(P, D, Q)_s$ is formulated as below:

$$W(t) = \sum_{i=1+s}^{P+s} \alpha_i W(t-i) + \sum_{i=s}^{Q+s} \beta_i e(t-i),$$

where $W(t) = X(t) - X(t - D * s)$.

The linear prediction model we use in this paper is a seasonal ARIMA model. Based on the periodic patterns observed Fig. 3, it seems appropriate to choose the seasonality parameter $s = 48$. To identify the right p, q parameters of the ARIMA model, auto-correlation functions (ACFs) and partial auto-correlation functions (PACFs) were plotted for all channels.

Fig. 4 illustrates the general tendency in the ACFs and PACFs calculates for all channels. Since the ACF is gradually decreasing while the PACF exhibits a sharp cut off after two hour lags, this is indicative of an AR signature present in the channel. The PACF drop when the lag is two indicates that the number of AR terms (*i.e.,* the p parameter) in the non-seasonal part is two. The ARIMA model which we are using is $(2, 0, 0)(2, 1, 0)_{48}$. The evaluation of ARIMA model is explained in details in Algorithm 1.

Algorithm 1 ARIMA based prediction

Input: SARIMAX model parameters $(p, d, q)(P, D, Q)_s$, number of steps to predict into the future n

1: **for** each channel **do**
2: split the data 80/20 into train set $(x_1, ..x_N)$ and test set $(y_1, .., y_M)$, number of iterations = M/n
3: **for** i:=1 **to** number of iterations **do**
4: fit the SARIMAX model to the training data
5: predict next n steps in the future, compare the predictions to first n steps of test set,(calculate Mean square error)
6: add the first n steps of test set to the training set = $(x_1, ..x_N, y_{i*n}, ..., y_{(i+1)*n})$, test set = $(y_{(i+1)*n+1}, ..y_M)$ and go to step 2
7: **end for**
8: **end for**

2) Time Delay Neural Networks: Neural networks (NN) are non-linear and non-parametric models composed of one *input* layer $\mathcal{X}$, one or multiple *hidden* layers $\mathcal{Z}_j$ and one *output* layer $\mathcal{Y}$. Each layer has a number of units denoted by x_i for the input layer, z_{ij} for the hidden layer(s) and y_i for the output layer. In the input layer, units correspond to input data whereas, in the hidden and output layers they represent a vector-to-scalar function. The topology or the architecture of a feed forward NN is defined by the number of units in each layer.

The input layer contains D units, each one corresponding to a different attribute in the data. The input units are represented as $x_i \in \mathcal{R}, i \in [1, D]$, and we set $x_0 = 1$ as the bias unit.

A sliding window method is used for framing the data set. Using the previous time steps to predict the next time step is called the sliding window method [22]. To determine the sliding window size, auto-correlation of the channels are used. Since a periodic behavior of 48 hours in most of the channels is observed in Fig. 3, we have set the lag value to 48. This means that the occupancy % of past 48 hours were used to predict the next occupancy % in future. Thus we have chosen the input layer with 48 units.

Weights are the connections between any two units and they

are denoted by $w_{ki}^{(l)}$ where i is the index for the source and k is the index for the destination unit, and j is the layer number. For example, the first hidden layer $j = 1$ contains H_1 units and the weights between the input and the hidden units are denoted by:

$$w_{ki}^{(1)} \in \mathcal{R}, \qquad i \in [0, D], k \in [1, H_1].$$

All of the weights in the model were initialized randomly close to 0. Each hidden unit applies a nonlinear transformation, through an *activation function*, to the weighted linear combination of the input units. The output of each hidden unit can be formulated as

$$z_{ij} = f(\boldsymbol{x}^{\boldsymbol{T}} \boldsymbol{w}_{\boldsymbol{i}}^{(\boldsymbol{j})}) \tag{2}$$

where z_{ij} is the i$^{\text{th}}$ unit of hidden layer j such that $i \in [1, H_j]$, $j \in [1, 2]$ $\boldsymbol{x}$ is the vector of inputs such that $(x_0, ..., x_D)$, $\boldsymbol{w}_{\boldsymbol{i}}^{(\boldsymbol{j})}$ is vector of weights for the i$^{\text{th}}$ hidden unit such that $(w_{i0}^{(j)}, ..., w_{iD}^{(j)})$. In this paper, we have used Rectified Linear Unit (ReLU) as our activation function in both hidden layers.

The output layer contains U output units and they are linear combinations of the hidden units with weights denoted by

$$w_{ki}^{(2)} \in \mathcal{R}, \qquad i \in [0, H_2], k \in [1, U],$$

where i is the index for the hidden and k is the index for the output unit. The hidden layer also has a bias unit such that $z_0 = +1$. Each output unit can be formulated as

$$\hat{y}_k = \boldsymbol{z}^{\boldsymbol{T}} \boldsymbol{w}_{\boldsymbol{k}}^{(\boldsymbol{2})}, \tag{3}$$

where $\hat{y_k}$ is the k$^{\text{th}}$ output unit such that $k \in [1, U]$, $\boldsymbol{z}$ is the vector of hidden units such that $(z_0, ..., z_{H_2})$, $\boldsymbol{w}_{\boldsymbol{k}}^{(\boldsymbol{2})}$ is vector of weights for the k$^{\text{th}}$ output unit such that $(w_{k0}^{(2)}, ..., w_{kH_2}^{(2)})$.

In training the TDNN, the back-propagation (BP) algorithm is used to compute the gradient of a loss function with respect to the weights of the network, w. Each observation in the training set can be represented as $(\boldsymbol{x}^{(\boldsymbol{n})}, y^{(n)})$ where $\boldsymbol{x}^{(n)}$ is the vector of input variables and $y^{(n)}$ is the target variable and $n \in [1, N]$ where N is the number of observations. The loss function we have used in this paper is the mean-square error and is given by:

$$J(w) = \frac{1}{n} \sum_{n=1}^{N} (\hat{y}^{(n)} - y^{(n)})^2. \tag{4}$$

We have used Adaptive Moment Estimation (Adam Algorithm) [23] as the optimizer for the loss function. We have opted for the default values of Adam optimizer in Keras version 2.2.2 which are the following: learning rate or $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Evaluation of the model was done by separating the data into train and test sets. A large portion of the data was used for training the model and the rest is used as test set to evaluate the performance of the model. In this paper, we have used the first 80% of time indices of each channel for training and the remaining 20% for testing.

Our proposed TDNN model, started with 2 hidden layers each containing 50 units. We have opted to increase the model capacity by favoring a network with fewer hidden layers and more hidden units in each one is a better fit than a network with more hidden layers and fewer hidden units. The model complexity was increased and its performance was investigated for signs of over-fitting. Increasing the model complexity was done by multiplying the number of units per hidden layer by a factor of two.

The user can change the frequency of the updates by determined a batch size. For example, if the batch size is N, then the weights are updated after the model goes over the whole training set. In this paper we are using a batch size of 30, meaning the model's weights are updated on the loss measured after observing 30 data instances.

Regularization methods such as early stopping, L2 regularization and dropout have been used in this respective order to tune the model's performance. Early stopping was used in training procedure, as a mean to prevent over-fitting. During model training, validation error was checked every $m = 5$ epochs to determine whether training should be stopped. The value of m is the patience argument in early stopping, meaning that if validation error has not decreased in the last m epochs, the training will be stopped.

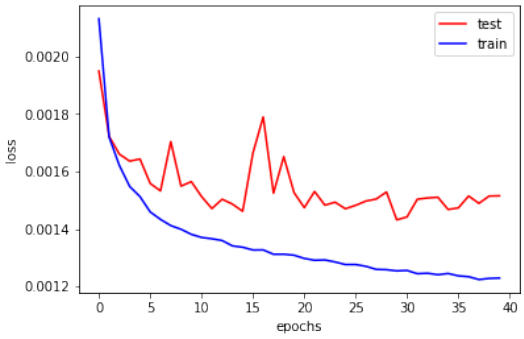

Fig. 5: MSE for the train and test loss versus epochs.

Fig. 5 illustrates the training and test loss during the training procedure of the model with 400 hidden units per layer and 40 epochs. One can observe that as the number of epochs is increasing, training error is decreasing. After epoch 30, it does not seem to improve the prediction performance on the validation set, indicating that the model has started to over-fit the training set. Therefore, we have determined that the appropriate number of epochs to use in training to be 30.

L2 regularization is implemented by modifying the Loss function and adding a new term which penalizes large weights. In this paper a L2-Regularization term is added to loss function, which is formulated as:

$$J(w) = \frac{1}{2} \sum_{n=1}^{N} (\hat{y}^{(n)} - y^{(n)})^2 + \lambda \sum_{j=1} {w_j}^2 \tag{5}$$

The training and testing procedure of TDNN and generating the loss for predictions is provided in Algorithm 2. The data in TDNN is a combination of all the channels that have been selected as described in Section III-A, and a single model is trained for all channels.

Algorithm 2 Training and testing procedure for TDNN

Input: train and test set, L2-regularization term λ, Dropout probability P_{drop}, early stopping patience m, number of epochs N, batch size B, number of steps to predict into the future $steps$

1: **for** i = 1 to $steps$ **do**
2: generate the model based on the input parameters, add Dropout and regularization terms
3: **while** early stopping criterion not met **do**
4: fit the model to the training data
5: **end while**
6: predict the i step into the future for the test set
7: calculate the mean square error y comparing the test set labels and predictions
8: **end for**

Increasing λ forces the weights towards zero, which results in a reduction of model's learning capacity. In this paper the effect of several values of λ have been tested on our model, which we provide in Fig. 6.

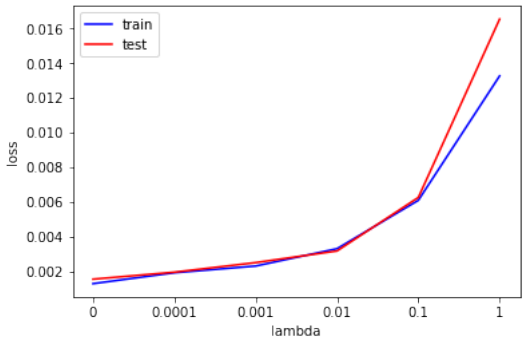

Fig. 6: MSE for the train and test loss versus values of λ

One can observe that for any value of $\lambda > 0$, the validation error starts to increase indicating the model does not benefit from L2-regularization. One possible explanation could be the use of early stopping, which in itself acts as a regularization technique, rendering L2-regularization redundant in calibrating the model.

Another approach is applying *dropout* to model layers, which randomly drops a defined probability of units along with their connections from the neural network during training [24]. For each training batch the network is re-adjusted and a new set of units are dropped out. At test time the weights are multiplied by the dropout probability of the associated units. This approach prevents model to be over-dependent on some hidden units. The dropout hyper-parameter which is usually between 0.2 and 0.5 and defines the probability that the weights associated with a particular unit will not be updated in training in a particular batch. Based on Fig. 7 increasing dropout is increasing the error, which means that adding dropout as a regularization factor is not effective.

IV. Results

We present the results in Fig. 8 and the numerical representation of the plots sampled every 5 steps in Table. I. Results illustrate that in predicting occupancy for the next

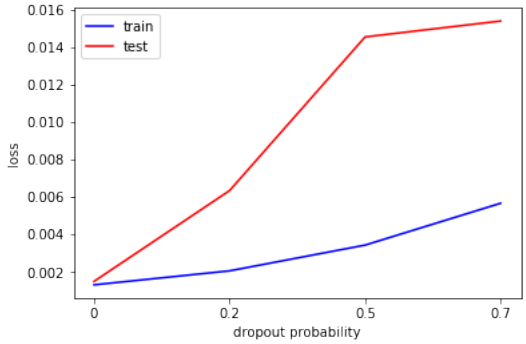

Fig. 7: MSE for the train and test loss versus values of dropout

50 time horizons, the TDNN has better or similar prediction performance to the the seasonal ARIMA model. We also provide the results for the baseline as it serves an indication on how much both models are able to improve upon a simple delay based guessing prediction regime.

Not surprisingly, the performance of all models gets worse as the prediction is made further into the future. The slight dip in the baseline model is due to the 48-hour periodic pattern we have observed in some channels.

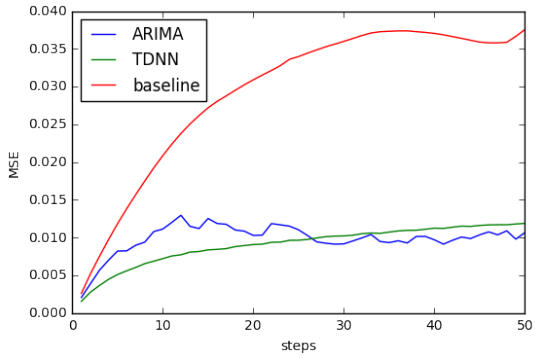

Fig. 8: Average performance (MSE) of three predictive models

While all models, including the baseline, start relatively similar in their performance ($\Delta t = 1$), the performance gain of the TDNN becomes pronounced as the prediction time horizon is increased slightly ($1 < \Delta t < 24$) before the gap between the seasonal ARIMA and TDNN based prediction performance closes. For further predictions into the future ($\Delta t > 24$), the prediction task becomes increasingly difficult and warrants further testing to understand how the performance of the ARIMA and the TDNN compare. This indicates that the non-linear nature of the TDNNs presents an improvement over linear prediction of the seasonal ARIMA. Furthermore, the TDNN can be trained offline for time horizons and can be used to provide offline prediction, whereas ARIMA model needs to be continuously trained in order to be able to forecast. The offline prediction is especially important in an IoT setting where the edge devices can simply query a server that will perform prediction that does not need a continuous stream of measurements from the environment.

TABLE I: RESULTS FOR AVERAGE MSE LOSS IN PREDICTIONS FOR DIFFERENT TIME HORIZONS

Model	$\Delta t = 1$	$\Delta t = 6$	$\Delta t = 11$	$\Delta t = 16$	$\Delta t = 21$	$\Delta t = 26$	$\Delta t = 31$	$\Delta t = 36$	$\Delta t = 41$	$\Delta t = 46$
Baseline	0.0025	0.0138	0.0224	0.0280	0.0315	0.0344	0.0364	0.0374	0.0369	0.0358
ARIMA	0.0020	0.0082	0.0120	0.0118	0.0103	0.0102	0.0095	0.0095	0.0091	0.0107
TDNN	0.0015	0.0056	0.0075	0.0084	0.0091	0.0097	0.0103	0.0109	0.0112	0.0116

A. Threats to validity

For validation, we randomly selected 70 channels from a list of 2684 channels to decrease the computational cost of the experiments. We are using 7 months of available continuous LMR data from October 2016 to April 2017 to validate the model. To tune the hyperparameters of the ARIMA model we have relied on ACF and PACF plots and the parameters have been chosen upon heuristic examination. The seasonal components of the ARIMA model has been selected based on the examination of periodic patterns that exits in some channels. On the other hand, the TDNN has been calibrated and regularized when it was able to start over-fitting but further capacity added to the model could improve these results. Finally, the training and testing procedures of the ARIMA and time delay neural networks exhibit some differences, due to the sliding windows forecasting method that accompanies ARIMA based predictions.

V. Conclusions and Future Work

In this paper we have built two predictive models, a seasonal ARIMA and a TDNN, and evaluated their performance in prediction the next 50 time steps into the future with respect to each other as well as a baseline method that simply used previous values as its prediction. The TDNN used in this paper is perhaps the simplest neural network variant designed to work with time series, and further performance gains may be achieved by training more sophisticated NN variants such as recurrent neural networks (RNNs) and in particular long short term memory networks (LSTMs) which also benefit from an added state that serves as the memory of such models. We aim to explore this direction further in future studies.

Acknowledgments

This research is supported in part by CRC Grant No: 50535.

References

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

[2] R. Bacchus, T. Taher, K. Zdunek, and D. Roberson, "Spectrum utilization study in support of dynamic spectrum access for public safety," in *DySPAN*. IEEE, 2010, pp. 1–11.

[3] B. Z. Kobb, *Wireless Spectrum Finder: Telecommunications,Government and Scientific Radio Frequency Allocations in the Us 30 MHz - 300 GHz*. New York, NY, USA: McGraw-Hill, Inc., 2001.

[4] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *JSAC*, vol. 23, no. 2, pp. 201–220, 2005.

[5] G. I. Alptekin and A. B. Bener, "Pricing model for the secondary market architecture in cognitive radio networks," in *Game Theory for Networks, 2009. GameNets' 09. International Conference on*. IEEE, 2009, pp. 479–483.

[6] ——, "An efficient spectrum management mechanism for cognitive radio networks," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*. IEEE, 2009, pp. 653–660.

[7] ——, "Customer oriented resource allocation framework in cognitive radio," *Computers & Industrial Engineering*, vol. 58, no. 3, pp. 401–410, 2010.

[8] E. Kavurmacioglu, M. Alanyali, and D. Starobinski, "Competition in private commons: price war or market sharing?" *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 29–42, 2016.

[9] Z. Wen, C. Fan, X. Zhang, Y. Wu, J. Zou, and J. Liu, "A learning spectrum hole prediction model for cognitive radio systems," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 2089–2093.

[10] 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, 2011.

[11] G. Ding, Y. Jiao, J. Wang, Y. Zou, Q. Wu, Y.-D. Yao, and L. Hanzo, "Spectrum inference in cognitive radio networks: Algorithms and applications," *IEEE Communications Surveys & Tutorials*, 2017.

[12] V. K. Tumuluru, P. Wang, and D. Niyato, "Channel status prediction for cognitive radio networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 10, pp. 862–874, 2012.

[13] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *INFOCOM*. IEEE, 2017, pp. 1–9.

[14] A. Gorcin, H. Celebi, K. A. Qaraqe, and H. Arslan, "An autoregressive approach for spectrum occupancy modeling and prediction based on synchronous measurements," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*. IEEE, 2011, pp. 705–709.

[15] J. Su and W. Wu, "Wireless spectrum prediction model based on time series analysis method," in *Proceedings of the 2009 ACM workshop on Cognitive radio networks*. ACM, 2009, pp. 61–66.

[16] Z. Lin, X. Jiang, L. Huang, and Y. Yao, "A energy prediction based spectrum sensing approach for cognitive radio networks," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*. IEEE, 2009, pp. 1–4.

[17] C. Stolojescu-Crisan, "Data mining based wireless network traffic forecasting," in *Electronics and Telecommunications (ISETC), 2012 10th International Symposium on*. IEEE, 2012, pp. 115–118.

[18] N. Baldo, B. R. Tamma, B. Manoj, R. R. Rao, and M. Zorzi, "A neural network based cognitive controller for dynamic channel selection," in *Communications, IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[19] H. Li, X. Xu, B. Wu, and X. Chen, "Multilayer feedforward neural network based efficient spectrum occupancy prediction scheme for cognitive radio system," *Journal of Computational Information Systems*, vol. 10, no. 10, pp. 4017–4028, 2014.

[20] L. Yin, S. Yin, W. Hong, and S. Li, "Spectrum behavior learning in cognitive radio based on artificial neural network," in *Military Communications Conference, 2011*. IEEE, 2011, pp. 25–30.

[21] M. I. Taj and M. Akil, "Cognitive radio spectrum evolution prediction using artificial neural networks based multivariate time series modelling," in *Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless), 11th European*. VDE, 2011, pp. 1–6.

[22] H. Hota, R. Handa, and A. Shrivas, "Time series data prediction using sliding window based rbf neural network," *International Journal of Computational Intelligence Research*, vol. 13, no. 5, pp. 1145–1156, 2017.

[23] J. d.P.Kingma, "Adam: a method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–15, 2015.

[24] A. I. R. N.Srivastava, G.Hinton, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.