A Machine Learning Based Scheme for Dynamic Spectrum Access

Anirudha Sahoo
Communications Technology Laboratory,
National Institute of Standards and Technology, Gaithersburg, Maryland, USA
anirudha.sahoo@nist.gov

***Abstract*—In this paper, we present a machine learning (ML) based dynamic spectrum access (DSA) scheme which can be used in a system in which the primary user (PU) spectrum occupancy can be represented as a sequence of busy (on) and idle (off) periods. We use real world data collected from Long Term Evolution (LTE) systems at two locations for our study. We experiment with different feed forward artificial neural network (ANN) architectures to choose from for our DSA scheme. A simple perceptron based ANN architecture was determined to provide good performance. We compare performance of our ML based DSA scheme with a traditional DSA scheme based on analytical model that uses survival analysis. Our results show that our ML based scheme outperforms the survival analysis based scheme in terms of utilization of idle periods. In terms of probability of interference to the PU, our scheme is better in some configurations and slightly worse in some other configurations.**

***Index Terms*—Dynamic spectrum access, spectrum sharing, machine learning, artificial neural network.**

I. Introduction

There is an acute scarcity of spectrum, especially below 6 GHz, due to static spectrum allocation policy. However, the spectrum utilization in some of the bands is low [1]. Thus, there is scope for increasing spectrum utilization in those bands through sharing. Hence, dynamic sprectrum access (DSA) has been proposed as a way of sharing spectrum with the incumbents to improve spectrum utilization. An incumbent is usually referred to as primary user (PU) and the user that shares the spectrum using DSA is referred to as secondary user (SU). Although there are different DSA schemes, in this study we focus on DSA based on opportunistic spectrum access (OSA). In an OSA based DSA system, an SU opportunistically transmits when the spectrum is idle due to inactivity of the PU. But the SU should vacate the spectrum before the PU reappears to avoid interfering with the PU. Since typically there is no communication between PU and SU, the SU has to predict when the PU might reappear, i.e., when the idle period might end. Based on this prediction, the SU decides how long it should transmit or if a request to transmit for a certain duration be granted. Thus, the performance of such a DSA system depends on how well the SU predicts the spectrum occupancy.

Machine learning (ML) has been used for various kinds of prediction problems, e.g., image recognition, natural language processing. So, it is only natural that ML-based techniques are used for DSA. Many DSA schemes have been designed based on analytical models. But analytical models have limitations. Normally, analytical models make certain assumptions to fit the DSA scenario to the model. These assumptions may not hold for certain datasets or for certain scenarios. Sometimes, there may be interdependencies in the input parameters which may not be adequately captured by the analytical models. These limitations can lead to performance degradation of the analytical models. ML based models, on the other hand, can overcome these limitations by having appropriate architecture to accommodate interdependencies of input parameters and by having enough training data in different scenarios to produce good results in many scenarios. Thus, in general, ML based models, with appropriate architecture, input features and training dataset can produce better results in many different scenarios. Although we found some ML-bsed schemes for DSA in the literature (see Section II), to the best of our knowledge, a very simple feed forward artificial neural network (ANN) based approach for DSA is missing. Hence, in this work, we devise a DSA scheme using basic feed forward ANN which can be used in a system in which spectrum occupancy of the PUs can be represented as a sequence of busy (on) and idle (off) periods. We used a Long Term Evolution (LTE) network as the PU system and considered the spectrum (channels) used in the uplink for DSA. For our study, we used real world LTE uplink data collected at two locations and devised an ANN based DSA scheme on a given channel of the LTE uplink. A channel in LTE is the smallest allocable range of frequency which is 180 kHz. We experimented with different input features for the ANN and decided on the set of input features that provided good performance. We also experimented with a few different ANN architectures and found that performance of a simple perceptron based ANN is comparable to that of more complex deep neural network (DNN) architectures. Hence, we chose a simple perceptron based ANN for our DSA scheme. The results from our experiments show that our ANN based scheme is able to utilize PU idle periods (whitespace) quite well with low probability of interference to the PU. We compared the performance of our ANN based scheme with a traditional DSA scheme based on an analytical model. The traditional DSA scheme, which is based on survival analysis, was propsed in [2]. From the experimental results, we observed that utilization of idle periods using our ANN based scheme was always higher than the survival ananlysis based scheme in all configrations. However, the probability of interference was lower for some configurations and slightly higher for some

other configurations. Therefore, this study shows that it is possible to design an effective DSA system using a simple feed forward ANN.

II. Related Work

There have been quite a bit of work reported in the literature on prediction of spectrum occupancy. A Partially Observable Markov Decision Process (POMDP) to predict spectrum occupancy has been proposed in [3]. The DSA scheme proposed in [4] uses expected remaining off time for prediction. Some schemes indirectly predict the spectrum occupancy by limiting the transmission duration of SUs based on some constraint. In [5], maximum bound on probability of interference to PU is used as a constraint to compute the duration of transmission of an SU. Spectrum occupancy is modelled as an alternating renewal process in [6] and residual idle time of the idle duration (in which SU request arrives) is used to indirectly predict when the spectrum will be busy again. Multiarm Bandit model has also been used in OSA channel access [7], [8]. Pattern mining of occupancy data has been used to predict channel idle period [9], [10]. DSA algorithms based on survival analysis have been presented in [2]. While [2] proposes channel access algorithms in time domain only, the survival analysis based algorithms presented in [11] are meant for DSA in time and frequency (or channel) domains.

Machine learning (ML) based techniques have been shown to be very effective with prediction in various fields (e.g., image recognition, natural language processing). It is no surprise that researchers have used ML to design DSA schemes. A Q-learning based algorithm to improve DSA performance in terms of channel throughput has been proposed in [12]. Faganello et al. proposed improvements to Q-learning algorithm to design three schemes for DSA in a dynamic industrial cognitive radio network [13]. Deep Q-learning was proposed as an ML approach by combining Q-learning and neural networks [14]. Such an architecture is called deep Q-learning network (DQN). In the DQN based algorithm proposed in [15], an SU avoids heavy interference regions and selects efficient frequency hopping pattern to dynamically access spectrum. In [16], a deep recurrent neural network is used to learn the time varying distribution of user traffic in a Land Mobile Radio (LMR) network, which is then used to determine best spectrum assignment and sharing strategies. In the graph neural network based approach for DSA in femtocells proposed in [17], the graph neural network maps the traffic load to a channel access scheme. In this work, a multiagent reinforcement learning framework is used for training and to predict channel quality.

III. Problem Formulation

In an opportunistic dynamic spectrum access (DSA) system, the spectrum occupancy by the PU can be represented as a series of *busy* and *idle* periods. An SU accesses the spectrum when it is idle and should finish transmission before the spectrum is occupied by the PU, i.e., before the next busy period starts. Since, it is not known exactly when the next busy period will start, the SU has to rely on some kind of prediction. Based on the accuracy of the prediction method, sometimes the SU may not finish transmission before the next busy period, which leads to interference with the PU. An efficient opportunistic DSA system keeps this interference to a low value.

Figure 1 shows an example scenario of opportunistic DSA. An SU request to transmit arrives at time instant 'A'. If the request is to transmit for duration τ_1 and the prediction system grants the request, then there will be no interference, i.e., the transmission will be successful. In this case the prediction system made the correct decision. On the other hand, if the requrest is to transmit for duration τ_2 and the prediction system grants the request, then there will be interference. In this case, the prediction system made a wrong decision; the correct decision is to deny the request. Note that in the first case, the prediction system could make a wrong decision and deny the request for transmission of duration τ_1, which would be a lost opportunity.

ML has been used in prediction systems [18], [19], [20]. Supervised learning using ANN is a good candidate for prediction. In such systems, an ANN is trained with a training dataset. The training dataset contains ground truth about the prediction for a set of inputs. For our problem, the training dataset contains some inputs for which the prediction is correct and some other inputs for which the prediction is wrong. Once the ANN is trained, it is used for prediction with input data which it has not seen before. However, for our application, we need to decide what should be the input features for the ANN. We also need to choose an ANN architecture which provides good performance. We make these choices by running experiments against different input featues and different ANN architectures.

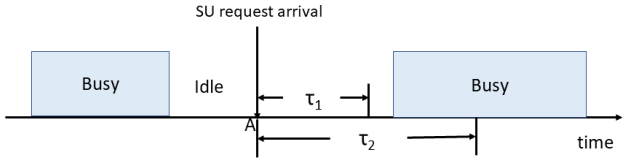

Fig. 1: Opportunistic DSA Example

IV. Evaluation

Our goal is to design a prediction system based on ANN that can facilitate opportunistic DSA for an SU. The ANN is trained using a training dataset. The training dataset consists of a set of inputs and the corresponding known output. In our case, the output is binary indicating whether the the SU request was granted or denied. Thus, our ANN based prediction system is a supervised binary classification system. Once the ANN is trained with the training dataset, when a request to transmit for duration τ arrives at an SU, the SU presents the input corresponding to the request to the ANN and gets an output. An output of 0 implies denying the request, whereas an output of 1 means the request is granted.

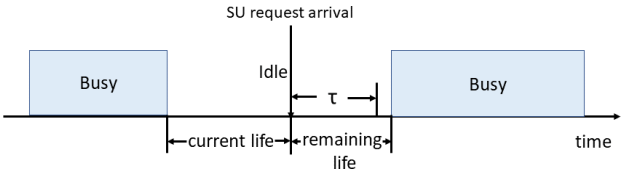

Fig. 2: Illustration of Some Input Features

A. Input Feature Selection

Selecting approporiate input features for an ANN is very important for its performance. In our case, given that the SU request arrives during an idle period, we know the *current life* of the idle period at the time of the request arrival. Current life of an idle period is the duration from the beginning of the idle period to the current time. Note that if we knew the *remaining life* of the idle period (the duration from current time until the end of the idle period), then we could deterministically decide whether to grant or deny the request which would eliminate the need for a prediction system. Hence, remaining life cannot be an input to the ANN. Figure 2 illustrates current life and remaining life of an idle period. Current life is undoubtedly an important feature for our ANN. Previous idle periods may also have correlation with the current idle period. Hence, we chose few previous idle periods as input features. Through experiments we found that ten previous idle periods is a good number. Lastly, the duration of SU request (τ) is also an important parameter for prediction, So, we chose current life of idle period, previous ten idle periods and the duration of SU request as our input features for the ANN. Thus, there were twelve nodes in the input layer.

B. Feature Scaling

We noticed that the values of the input features were quite varied, i.e., while some were small in value, others were very large. Hence, the values of input features were normalized to a value between 1 and 100.

C. Data Collection

For our experiements, we collected real world LTE data at two locations.

First set of data was collected indoors inside one of the labs at the National Institute of Standards and Technology (NIST). An Ettus Universal Sofware Radio Peripheral (USRP) [1] running USRP hardware driver (UHD) version 003.009.001 was fitted with a small 10.78 cm rubber duck antenna. Using GNU Radio version 3.7.9rc1, complex-valued I/Q samples were collected every 80 ns with a sampling rate of 12.5 MHz in the Band 17 with a 10 MHz uplink (UL) LTE band at center frequency 709 MHz. In every 50 μs period, 625 consecutive I/Q samples were collected and power spectrum over that period was computed. Average power spectra over 20 consecutive periods (equal to 1 ms) were then computed. 56 point power spectrum for each 1 ms period was then generated by binning the coefficients. Each power spectrum coefficient was rounded to the nearest integer and represented as an 8 bit integer. Each of these coefficients represents power (in dB) over a 180 kHz frequency range. 50 LTE channels in a 10 MHz UL are represented by the middle 50 coefficients. These power values were then converted to binary on or off (0 or 1) by applying a noise threshold to each of the 50 channels. The noise threshold was determined as follows. Samples were collected for a one hour period from the USRP after connecting a matched-load terminator to the receiver port. The noise threshold was determined to be the level at which 1 % of the sample values were above the threshold. This corresponded to probability of false alarm (PFA) of 1 % [21]. We collected the data on two week days at the same time of the day. The idea is to use one dataset as training data and the other as test data. The first dataset was collected on a Monday from 3 PM to 4 PM local time, whereas the second dataset was collected on the nextday (Tuesday) from 3 PM to 4 PM. These timings were chosen assuming that there would be high LTE traffic during those times.

The second dataset was collected in the Philadelphia metropolitan area near University of Pennsylvania (we will refer to this as *UPENN* data). We requested the administrator of the CityScape spectrum [22] to collect LTE uplink spectrum data. LTE uplink data was collected on a weekday at 1:17 PM local time for an hour. Additional processing of the I/Q samples was carried out to bring the data to the same format as the data collected at NIST. Finally, the CityScape datasets was converted to binary occupancy sequences using the noise threshold recommended by the CityScape administrator. The noise threshold at the CityScape site was determined by measuring the power level when no UE was believed to be communicating. The one hour worth of data is split into two equal parts of half an hour each. One part was used as training data whereas the other was used as test data.

D. Generation of Training and Testing Dataset

Given the spectrum occupancy data, the training dataset was created as follows. SU requests were simulated to arrive with a Poisson distribution, i.e., the interarrival times of the requests were exponentially distributed with a given mean arrival rate. Experiments were repeated with different mean arrival rate. For a given SU request, the requested transmission time was uniform randomly chosen between 1 ms and 10 ms. Since the arrival time of the SU request was known, current life of the idle period was computed and then the remaining life (see Figure 2) of the current idle period was computed. The last ten idle periods, current life of this idle period and the SU request duration form an input set. If this remaining idle duration was greater than the SU's requested transmission time, then the transmission was successful and the output label for this input set was set to 1, otherwise it was set to 0. The training data was randomly shuffled and then split into two equal halves. The first half was used to train the ANN and the second half was used as validation data.

[1]The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

The testing dataset was built exactly with the same manner, but on a different collected dataset. For example, for datasets collected at NIST, dataset of one day was used to generate training data whereas dataset of the other day was used to generate testing data. Similarly for the UPENN dataset, a half hour dataset was used to generate training data whereas the other half hour dataset was used to generate test data.

E. Choosing an Appropriate ANN Architecture

We need to decide what kind of ANN architecture and configuration are appropriate for our application. Towards that goal, we used the binary occupancy data of channel 5 of the NIST dataset. The first day's data was used for training and the second day's data was used for testing. Table I lists the different architectures and the corresponding results. We used keras on a server running linux to implement our code. For all architectures, the training was stopped after 100 epochs. Reducing the number of neurons from one hidden layer to the next (as we move towards the output layer) has been reported to be a good rule of thumb for ANN architecture [23]. Hence, we started with a Deep Neural Network based ANN architecture with three hidden layers, the first with 20 nodes, the second with 10 and third with 5 nodes. The training accuracy with this architecture was 78.47 %, whereas validtion accuracy was 77.57 %. Testing accuracy, in this case, was 91.8 %. We then simplified the architecture by having only two hidden layers. In this case, the training and validation accuracy came down only a little, but testing accuracy increased to 95.9 %. So, we continued to simplify the architecture as shown in the table. When we reduced the number of hidden layers and also reduced the number of nodes in the hidden layer, we did not see any difference in the testing accuracy, whereas the training and validation accuracy decreased slightly. So, we went all the way down to no hidden layer, i.e., just a single perceptron. Even with a single perceptron, the training and validation accuracy went down by a very small amount, whereas the testing accuracy remained the same. Hence, a single perceptron based architecture was deemed appropriate for our application. So, performance evaluation of our scheme was done with a single perceptron architecture as shown in Figure 3. Once trained, this ANN architecture can predict whether an SU request should be accepted or not in O(N) time, where N is the number of inputs to the perceptron. Therefore, run time of our DSA scheme is low with just twelve inputs.

F. Notations Used

As mentioned earlier, for the NIST data, data collected on one day was used for training whereas the data collected on the other day was used for testing. When NIST data is used in our experiments, we denote a configuration as NIST_*train_test*, where *train* and *test* are either *day1* or *day2* depending on which day's data is used for training and which day's data is used for testing. For example, configuration *NIST_day1_day2* implies day1 data is used for training the perceptron and day2 data is used for testing. Similarly, for data collected at UPENN, the configuration is denoted as UPENN_*train_test*, where *train* and *test* are either *hh1* or *hh2*, depending on whether first half hour (hh) or second half hour data is used for training. For example, configuration *UPENN_hh2_hh1* represents UPENN data collected in the seocnd half hour used as training data, whereas data collected in the first half hour used as testing data.

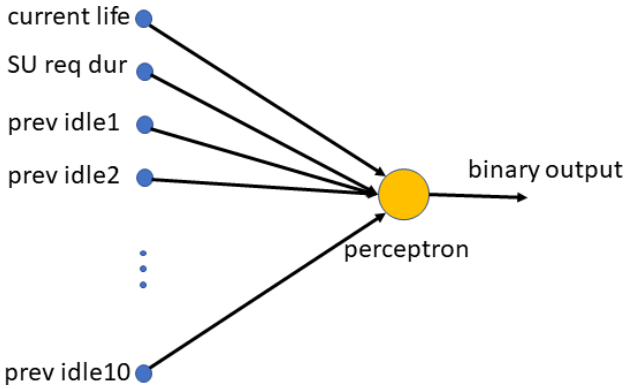

Fig. 3: Final Architecture adopted for our application. It consists of twelve node input layer connected to a single perceptron with a binay output. The perceptron uses *sigmoid* activation function, *binary cross entropy* loss function and *adam with Nesterov momentum (nadam)* optimizer.

In an earlier work, we proposed an analytical model based on *survival analysis* to design a DSA system [2]. Essentially, it computes a non-parametric estimate of survival function [24] from the idle time distribution of occupancy data. It then computes an estimate of cumulative hazard function. Given an upper bound on probablity of successful transmission, it formulates an approximate test statistic based on difference of cumulative hazard function at two different time instants (see Eqn (14) in [2]). This equation becomes the basis of two prediction algorithms (predict whether a requested SU transmission will be successful or not). We compare the performance of our ANN-based scheme, (henceforth refered to as DSA-ML scheme), with the survival analysis based scheme based on Algorithm 1 (we call DSA-SA scheme) described in [2].

G. Metrics

The following two metrics were used to evaluate the performance of DSA-ML scheme and also to compare the performance of DSA-ML scheme with DSA-SA scheme.

- White Space Utilization (WSU): White Space Utilization (WSU) of an SU on a given channel is the fraction of total idle time (or white space) used by the SU to transmit its data. It is the ratio of total idle duration used for transmission by an SU on a given channel to the total idle duration on that channel.
- Probability of Interference (PoI): The Probability of Interference (PoI) of an SU for a given channel is defined as the probability of that the SU's tranmission collides with a PU transmission. Hence, it can be approximited to be the ratio of the number of collisions with the PU

ANN Architecture	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss	Test Accuracy (%)
three hidden layers, first with 20 nodes, second with 10 nodes, and third with 5 nodes	78.47	0.441	77.57	0.455	91.8
two hidden layers, first with 10 nodes and second with 5 nodes	77.41	0.456	77.14	0.461	95.9
one hidden layer with 8 nodes	76.95	0.463	75.93	0.469	95.9
one hidden layer with 4 nodes	76.82	0.468	75.46	0.481	95.9
one hidden layer with 2 nodes	76.36	0.479	76.51	0.478	95.9
no hidden layer (single perceptron)	74.96	0.500	74.99	0.500	95.9

TABLE I: Experiment results with different ANN architectures. Hidden layers were fully connected and used *RELU* activation function, whereas the output node used *sigmoid* activation function. *binary cross entropy* loss function and *adam with Nesterov momentum (nadam)* optimizer were employed in the ANNs. Training was stopped after 100 epochs.

	NIST Dataset		UPENN Dataset	
	day1	day2	hh1	hh2
mean idle dur (ms)	80.11	45.31	10.78	13.37
stdev idle dur (ms)	373.58	97.60	10.78	13.39
% of time channel idle	98.69	97.75	90.51	92.41

TABLE II: Some statistics of idle duration in the Datasets

transmissions to the total number of SU transmission over a very long observation period.

V. Results

In this section we present results of our experiments. We have two sets of results, one using NIST dataset and the other using UPENN dataset. In all our experiments, the SU requests arrive with a Poisson distribution and the mean interarrival time of the request is varied. Every SU request is for transmitting for a constant duration of 2 ms (i.e., $\tau = 2$ ms). For NIST dataset we used the data corresponding to channel 5, whereas for UPENN data we use the data of channel 21. Some statistics of idle durations for the two datasets are provided in Table II.

A. Performance Evaluation using NIST Dataset

Figure 4 compares WSU of DSA-ML and DSA-SA schemes as mean interarrival time of SU request increases. In this case, the occupancy data of first day (day1) was used for training (for both DSA-ML and DSA-SA schemes) whereas the data of second day (day2) was used for testing in DSA-ML scheme and for running Algorithm1 in DSA-SA scheme, i.e., this uses configuration NIST_day1_day2. For both the schemes, WSU decreases as the mean SU request inter-arrival time increases. As mean SU request inter-arrival time increases, the number of SU request in the observation period decreases. Since the duration of SU transmission is constant (2 ms), the amount of white space utilized for SU transmission decreases. We also notice that WSU for DSA-ML scheme is always higher than that of DSA-SA scheme.

Figure 5 depicts the same performance comparision as Figure 4, except that day2 data was used for training whereas day1 data was used for testing (configuration NIST_day2_day1). In this case also we see very similar results. WSU for DSA-ML scheme is always higher than that of DSA-SA scheme. So, DSA-ML scheme outperforms DSA-SA scheme in both the configurations. DSA-SA scheme assumes that the channel idle times are independent and uses a non-parametric estimate of cumulative hazard function in its algorithms. This assumption of independence of idle times and approximation of cumulative hazard function adversely affect its performance. DSA-ML scheme is able to overcome these limitations to some extent by training its neuron on the large training dataset. Hence, DSA-ML scheme performs better than DSA-SA scheme. WSU for both the schemes do not change much between NIST_day1_day2 and NIST_day2_day1 configurations. This signifies that both the DSA schemes are robust against changing training dataset.

Figures 6 and 7 show the variation of PoI as mean SU request inter-arrival time increases when configurations NIST_day1_day2 and NIST_day2_day1 are used respectively. In configuration NIST_day1_day2, PoI for DSA-ML scheme is slightly lower than that of DSA-SA scheme and for both the schemes the PoI does not vary much as mean SU request inter-arrival time increases. This is a useful property which implies that the interference to the PUs will not vary much when the SU request arrival rate changes. The PoI for DSA-ML scheme is 0.04 which is pretty low. With NIST_day2_day1, PoI is even lower for both the schemes and does not vary much with increasing mean SU request inter-arrival time. Although PoI for DSA-ML scheme is slightly higher than that of DSA-SA scheme, it is only 0.023, which is very low.

It is worth noting that at mean SU request inter-arrival time of 2 ms the WSU for DSA-ML scheme for either configuration is above 45 % while incurring very low PoI (0.041 for NIST_day1_day2 configuration and 0.023 for NIST_day2_day1 configuration). This indicates that our DSA-

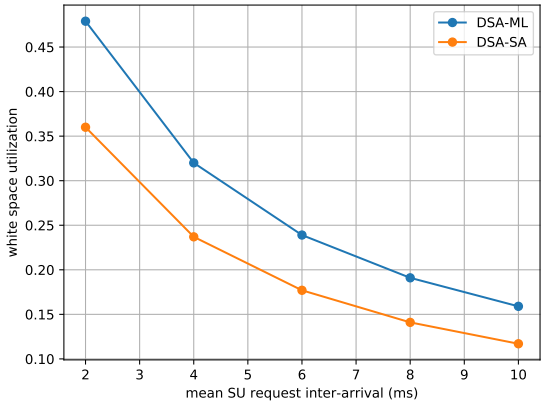

Fig. 4: WSU vs inter-arrival time (config NIST_day1_day2)

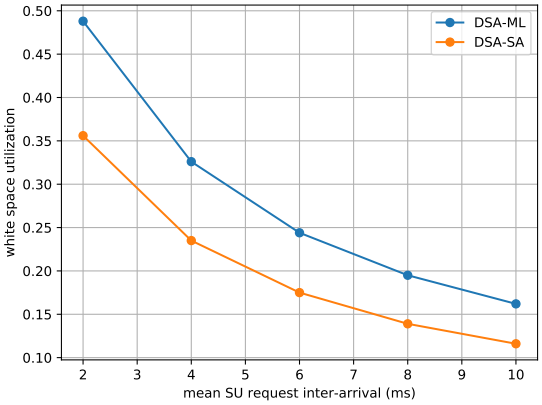

Fig. 5: WSU vs inter-arrival time (config NIST_day2_day1)

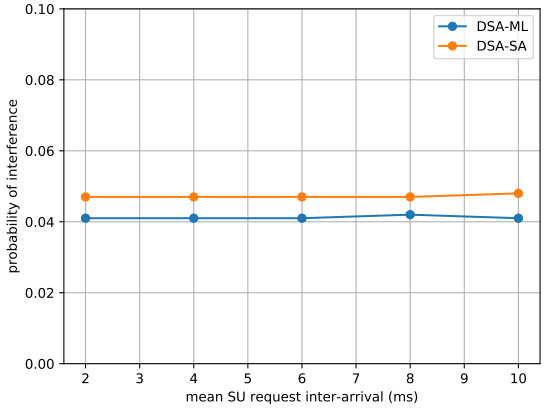

Fig. 6: PoI vs inter-arrival time (config NIST_day1_day2)

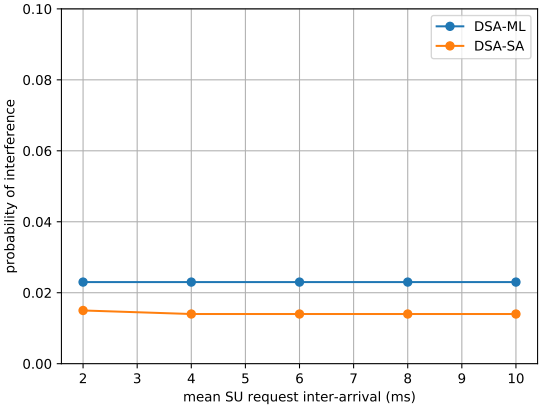

Fig. 7: PoI vs inter-arrival time (config NIST_day2_day1)

ML scheme can be used to achieve reasonably high WSU with a low PoI.

B. Performance Evaluation using UPENN Dataset

There is no performace numbers available for DSA-SA scheme with UPENN dataset. Hence, for this dataset we only present the results with DSA-ML scheme. Figure 8 shows the variation of WSU as mean SU request inter-arrival time increases for upenn_hh1_hh2 and upenn_hh2_hh1 configuration. As expected, WSU decreases as mean SU request inter-arrival time increases. We notice very little difference between WSU values across the two configurations, which again shows the robustness of the DSA-ML scheme for this dataset. Figure 9 shows the variation of PoI as mean SU request inter-arrival time increases for upenn_hh1_hh2 and upenn_hh2_hh1 configurations. For both the configurations PoI remains almost constant.

VI. Conclusions and Future Work

We devised a DSA scheme based on feed forward ANN which can be used in a system in which spectrum occupancy of PUs can be modelled as a sequence of busy and idle periods. We experimented with different feed forward ANN architectures for opportunistic DSA using real world LTE uplink data. Our experiments showed that a simple perceptron based ANN produces good performance for the DSA scheme. The dataset has high spectrum availability, i.e., percentage of idle duration was high and the SU request was for short duration (2 ms). So, we believe that a simple ANN architecture is adequate when spectrum has high availability and SU requests are relatively short. We compared our ML based scheme (DSA-ML) with a traditional analytical model based scheme (DSA-SA) that uses survival analysis. Our DSA-ML model performed better than DSA-SA model in terms of WSU in all the configurations. In terms of PoI, DSA-ML model sometimes performed better and some other times it was slightly worse than the DSA-SA model.

In terms of future work, we want to experiment with different ANN architectures when the spectrum availability may not be as high as in our collected dataset. For that, we may have to generate synthetic data. We also want to try a support vector machine (SVM) based model with the collected dataset

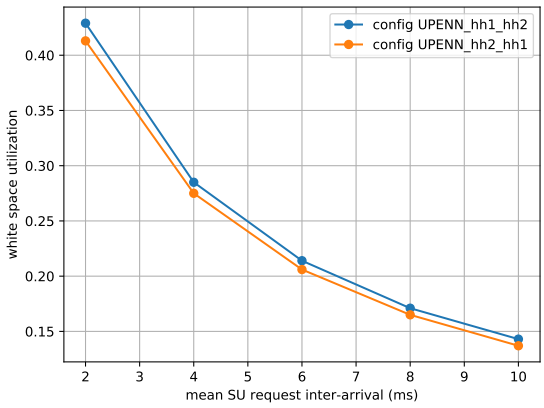

Fig. 8: WSU vs inter-arrival time using UPENN dataset in DSA-ML Scheme

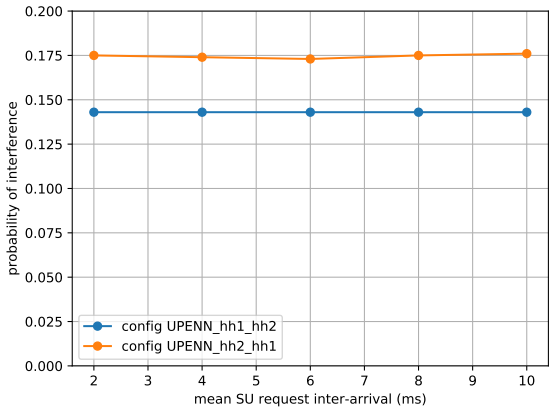

Fig. 9: PoI vs inter-arrival time using UPENN dataset in DSA-ML Scheme

and observe its performance. In this work, an SU request was only in one dimension: time. We would like to extend it to serve SU request in two dimensions: time and frequency. This, in LTE parlance, is a request for physical resource block (PRB). The SU would request for a certain number of PRBs and we need to build an ANN which can predict whether to grant or deny the request. If the request is granted, then the ANN should provide the map of the allocated PRBs.

REFERENCES

[1] K. S. Tugba Erpek and D. Jones, *Dublin Ireland Spectrum Occupancy Measurements Collected On April 16-18, 2007*, 2007 (accessed September 21, 2020), http://www.sharedspectrum.com/wp-content/uploads/Ireland_Spectrum_Occupancy_Measurements_v2.pdf.

[2] T. A. Hall, A. Sahoo, C. Hagwood, and S. Streett, "Dynamic spectrum access algorithms based on survival analysis," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 740–751, Dec 2017.

[3] Q. Zhao, L. Tong, A. Swami and Y. Chen, "Decentralized Cognitive MAC for Opportunistic Spectrum Access in Ad Hoc Networks: A POMDP Framework," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 2, pp. 589–600, April 2007.

[4] K. W. Sung, S. Kim and J. Zander, "Temporal Spectrum Sharing Based on Primary User Activity Prediction," *IEEE Transactions on Wireless Communications*, vol. 9, no. 12, pp. 3848–3855, December 2010.

[5] A. Plummer, M. Taghizadeh and S. Biswas, "Measurement based bandwidth scavenging in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 19–32, January 2012.

[6] M. Sharma and A. Sahoo, "Stochastic Model Based Opportunistic Channel Access in Dynamic Spectrum Access networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1625–1639, July 2014.

[7] C. Tekin and M. Liu, "Online learning of rested and restless bandits," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5588–5611, August 2012.

[8] Y. Gai and B. Krishnamachari, "Decentralized online learning algorithms for opportunistic spectrum access," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, December 2011, pp. 1–6.

[9] S. Yin, D. Chen, Q.Zhang, M. Liu and S. Li, "Mining Spectrum Usage Data: A Large-Scale Spectrum Measurement Study," *IEEE Transactions on Mobile Computing*, vol. 11, no. 6, pp. 1033–1046, June 2012.

[10] P. Huang, C-J. Liu, X. Yang, L. Xiao and J. Chen, "Wireless Spectrum Occupancy Prediction Based on Partial Periodic Pattern Matching," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1925–1934, July 2014.

[11] A. Sahoo, T. A. Hall, and C. Hagwood, "Optimal dynamic spectrum access scheme for utilizing white space in lte systems," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–8.

[12] C. Lv, J. Wang, F. Yu, and H. Dai, "A q-learning-based dynamic spectrum allocation algorithm," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. Atlantis Press, 2013.

[13] L. R. Faganello, R. Kunst, C. B. Both, L. Z. Granville, and J. Rochol, "Improving reinforcement learning algorithms for dynamic spectrum allocation in cognitive sensor networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 35–40.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] G. Han, L. Xiao, and H. V. Poor, "Two-dimensional anti-jamming communication based on deep reinforcement learning," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2087–2091.

[16] H. Rutagemwa, A. Ghasemi, and S. Liu, "Dynamic spectrum assignment for land mobile radio with deep recurrent neural networks," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[17] H. Jiang, H. He, and L. Liu, "Dynamic spectrum access for femtocell networks: A graph neural network based learning approach," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 927–931.

[18] Y.-F. Li, H.-W. Zha, and Z.-H. Zhou, "Learning safe prediction for semi-supervised regression," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[19] M. D. Dyer, T. Murali, and B. W. Sobral, "Supervised learning and prediction of physical interactions between human and hiv proteins," *Infection, Genetics and Evolution*, vol. 11, no. 5, pp. 917–923, 2011.

[20] R. Caromi and M. Souryal, "Detection of incumbent radar in the 3.5 ghz cbrs band using support vector machines," in *2019 Sensor Signal Processing for Defence Conference (SSPD)*. IEEE, 2019, pp. 1–5.

[21] M. Lopez-Benitez and F. Casadevall, "Methodological aspects of spectrum occupancy evaluation in the context of cognitive radio," in *2009 European Wireless Conference*, May 2009, pp. 199–204.

[22] S. Roy, K. Shin, A. Ashok, M. McHenry, G. Vigil, S. Kannam, and D. Aragon, "Cityscape: A metro-area spectrum observatory," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, pp. 1–9.

[23] *Why is it common in Neural Network to have a decreasing number of neurons as the Network becomes deeper*, 2017 (accessed August 15, 2020), https://www.quora.com/Why-is-it-common-in-Neural-Network-to-have-a-decreasing-number-of-neurons-as-the-Network-becomes-deeper.

[24] R. G. Miller Jr, *Survival analysis*. John Wiley & Sons, 2011, vol. 66.