FederationS: Federated Learning for Spatial Reuse in a multi-BSS scenario

Jernej Hribar and Andrea Bonfante

The authors are with CONNECT Centre, Trinity College Dublin, Ireland
email: {jhribar,bonfanta}@tcd.ie

***Abstract*—We are all aware that Wi-Fi's Achilles heel is its poor performance in crowded scenarios such as airports, stadiums and public gatherings. The recent IEEE 802.11ax amendment includes spatial reuse (SR) to increase simultaneous transmissions and improve Wi-Fi performance in these situations. However, one of the main limitations of the SR mechanism is that it relies on local information, limiting the effectiveness of this technique. This report proposes a distributed method based on Federated Learning (FL), which enables the selection of the best Overlapping Basic Service Set (OBSS)-Preamble Detection (PD) configuration by predicting the achievable throughput and without transferring data to a central server. First, we train the FL model with synthetic data generated by a standards-compliant system-level simulator. Then, we discuss the results and the lessons learned during this experience.**

I. Introduction

In this report, we propose applying FL to predict the station (STA)'s throughput for Spatial Reuse (SR) operations in the IEEE 802.11ax (11ax) system [1]. The key technique to improve SR relies on setting the OBSS-PD threshold to increase the number of concurrent transmissions in a multi-Basic Service Set (BSS) scenario. Differently from state-of-the-art works that propose to control the OBSS-PD configuration based on local information and heuristic approaches [2], we propose to select the optimal configuration of the OBSS-PD threshold in a distributed manner with FL. Since its inception in 2016 [3], FL proved to be a very effective Machine Learning (ML) technique that can be employed as an alternative to the conventional ML-based solutions, which usually entail centralised training. This is because constraints such as the massive amount of data to transfer from the edge to a central server and stringent General Data Protection Regulation (GDPR) policies for data privacy prevent data transfer to the central server, making centralised ML techniques challenging to be applied in practice. Moreover, in a multi-operator setting the sharing of data between BSS of different vendors may reveal sensible information and disclose proprietary design or techniques. Instead, the FL approach performs the training on the edge device. Only the result of the training, namely the model parameters, are communicated to the central server.

Fig. 1 shows the system model consists of K contexts and a central server. We define a set $\mathbb{K} = \{k_1, k_2, \ldots, k_K\}$ that includes the K contexts. Each context k is formed by a different deployment of Wireless Local Area Network (WLAN) devices, and it is characterised by different properties such as Access Point (AP) and STA devices locations, AP load and interfering conditions. In each context a reference BSS tests different OBSS/PD configurations in the range $\{-82, -81, \ldots, -62\}$ dBm with 1 dBm step. At the same time, other BSSs keep their OBSS/PD configuration fixed to the default value of -82 dBm. We leverage the FL approach to perform training at the AP solely based on the data acquired in the IEEE 802.11ax context. Statistics and parameters are organised in a local dataset $\mathcal{P}_k$ with related input and output variables $\{x_k, y_k\}$ that are used for training the model at the edge. At time instant t, each context communicates the updates of the local model $w_{t,k}$ to a central server, which collects the updates from all the contexts and computes the aggregated model. In a subsequent time instant $t+1$, the central server communicates back the global model update w_{t+1} to all the contexts. The same process is repeated for several communications rounds until the solution convergence.

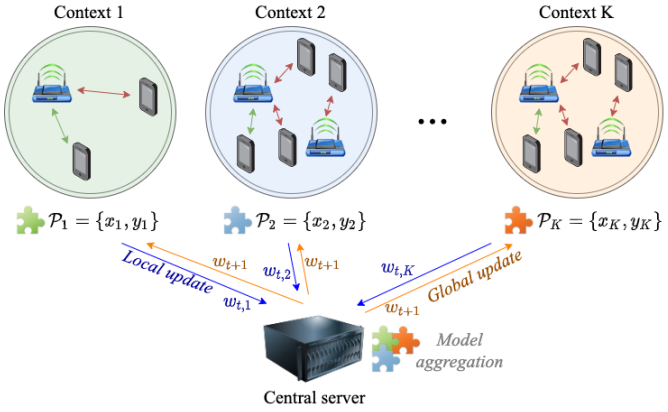

Fig. 1: Representation of the FL system with different contexts formed by different APs and STAs. Each context trains the ML model with local data and sends the update to a central server that communicates the global update after model aggregation.

The rest of the report is organised as follows. In Sec. II, we describe the prepossessing data operations. In Sec. III, we describe the Deep Neural Network (DNN) model running in each client. In Sec. IV, we describe the proposed FL solution, and we describe the numerical results. Finally, in Sec. V, we document the lessons learned from this experience.

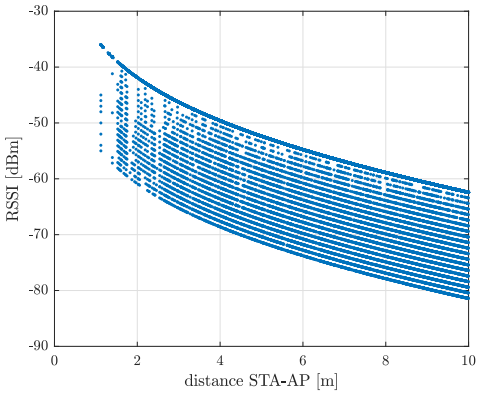

Fig. 2: Representation of the RSSI reported by the STA to the AP as a function of the distance between STA and AP for different OBSS/PD configurations.

II. Data Analysis

In this part, we discuss how features are selected for the training process. We start extracting the set of features available in the simulator's output file of each context, namely the OBSS/PD configuration, the Received Signal Strength Indicator (RSSI), the set of interference powers from other APs sensed by the AP in the reference BSS, the signal to interference noise ratio (SINR) and throughput for each STA served by the AP. Then, we consider the additional information available in the simulator's input file of each context. Here, we extract the coordinates of the AP and the ones of the STA. Then, we compute the Euclidean distances from each STA to the serving AP. Moreover, we compute the number of STA served by the AP and the number of APs interfering with the server AP in the BSS under analysis. It is worth noting that the RSSI, SINR and throughput measurements can be reported periodically by each STAs. Interference powers can be measured during the listen-before-transmit (LBT) phase at the AP that may employ Multiple-Input-Multiple-Output (MIMO) Receiver (Rx) processing techniques to separate the different interfering sources. In addition, the distance between STA and AP can be determined with Time-of-Arrival (TOA) ranging techniques.

To process the data we clean the input and output data removing all non-numerical values from the dataset. Then, we arrange the data of each STA to form 1-D vectors with 11 numerical entries used as the input of the model and containing all measurements and system parameters. Conversely, we define the STA throughput as the target variable and output of the model.

To note that the features present entirely different ranges between maximum and minimum values and are expressed with different units of measurements, e.g. dBm for RSSI and interference power, dB for SINR and meters for distances. To balance each feature's contribution to the overall model predictions, we re-scale the features with the Min-Max normalisation method that transforms all features' in the range $[0, 1]$. Finally, when input data are missing, like when the number of interfering APs reported is less than the five recorded according to the system parameters, we assign those values with 0s. The activation function that we will explain later is chosen to keep neurons inactive when 0s are present at the input.

Secondly, we conduct a more careful inspection of the data by analysing Fig. 2, which shows the attenuation of the RSSI increasing the distance between STA and AP for each configuration of OBSS/PD threshold, i.e. $\{-82, -81, \ldots, -62\}$ dBm, in ascending order from top to bottom. The figure shows that higher OBSS/PD thresholds correspond to lower RSSI received at a given distance. This is a result of the OBSS/PD mechanism implemented in the system-level simulator [4]. The AP reduces the Transmitter (Tx) power when using the more aggressive OBSS/PD threshold configuration, leading to a higher probability of accessing the channel due to the limited range for sensing other BSS transmission [5]. Therefore, as a design choice in the data pre-processing, we change the signs of the input data corresponding to both distance and OBSS/PD threshold features. This makes the RSSI, distance, and OBSS/PD input data move in the same direction and be positively correlated.

The results of data pre-processing are shown in Fig. 3, where we represent the correlation matrix between input and output variables. Correlation values close to 0 indicate a lack of relations and structure between the data corresponding to these variables, while correlation values close to -1 and $+1$ indicate a perfect negative and positive correlation between variables, respectively. Looking to Fig. 3 two main observations can be made:

1) Most features show a positive or negative correlation with the output variable (throughput). As expected, only the OBSS/PD feature does not directly affect the throughput. Otherwise, the problem would be trivial to model. Indeed, the OBSS/PD value is correlated to the RSSI as discussed before, which affects SINR and throughput variables, showing that relationships between input and output values of the system are not straightforward to discover and to characterise with domain models. This justifies the adoption of DNN, which are extensively used for their capabilities to model nonlinear relationships.
2) Two regions depicted with lighter colours at the top left and at the bottom right of the correlation matrix identify two groups of features that show a strong positive correlation between input variables. Thus, in the DNN architecture design, these inputs of the model need to be fully connected. In contrast, the two regions at the top right and bottom left are characterised by elements with close to zero correlation values, meaning that the relationship between features is not strong. Therefore, these connections are expected to bring a low contribution to the predictions and can be dropped in the DNN architecture design.

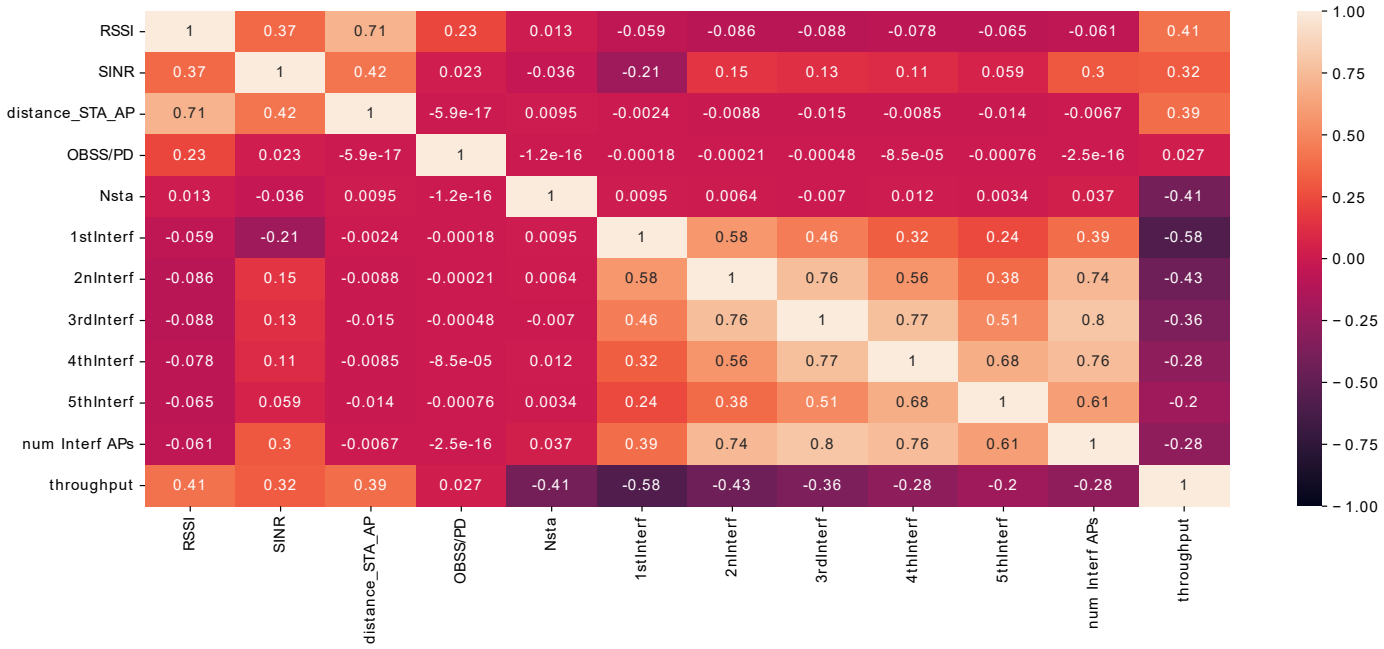

Fig. 3: Representation of correlation matrix showing the correlation between different input and output variables of the dataset.

III. DNN Model Design

Based on the observations highlighted in Sec. II, we design the model architecture represented in Fig. 4. First, we split the DNN model into two parallel branches. The inputs of the first branch are the features constituting the first block, i.e. RSSI, SINR, distance STA-AP and OBSS-PD threshold. At the same time, features like the number of STA, the interfering AP powers and the number of interfering APs form the second block of features and are used as input of the second branch. The input layers are followed by two hidden layers defined for each branch separately. We use a concatenation layer to merge the output of these two branches.

The result of the concatenation is then used as input of two additional hidden layers, which are connected to the output layer of the model. We adopt the tanh activation function to provide positive and negative outputs and keep neurons inactive when the inputs are 0s. Finally, we add dropout layers after each layer before the output layer to reduce overfitting.

In Fig. 5, we report the results of the training with a centralised model, which takes as input the data from all the contexts. This represents the lower bound achievable by the the FL solution, and guide us towards building the FL solution described in the next section, as we use the same DNN architecture in each client.

IV. Federated Solution

Our FL a solution is based on the implementation outlined in [6]. However, in our approach, the central server combines the trained weights in an unique way to capture the scenario specific challenges of the spatial reuse in multi-BSS system.

Algorithm 1 Proposed Federated Learning Solution.

1: Initialise set $\mathbb{K}$,, i.e., init K contexts with data samples
2: From $\mathbb{K}$, select N_{eval} to create $\mathbb{K}_{val}$
3: Create a new set of contexts $\mathbb{K}_{tr} = \mathbb{K} \cap \mathbb{K}_{val}$
4: Server initialises model parameters θ^0 and W^0
5: The server transmits θ_k^0, W_k^0 to k-th contexts
6: **for** communication epoch$t = 1, T$ **do**
7: Randomly select N_{tr} contexts from $\mathbb{K}_{tr}$ to get $\mathbb{K}_{ep}$
8: **for** i-th context in $\mathbb{K}_{ep}$ **do**
9: The i-th context updates the model using
10: Split context's samples in β ($\frac{n_i}{B}$ batches of size B)
11: Where n_i represents the number of data samples
12: **for** batch b in β **do**
13: $\theta_i^t, W_i^t \leftarrow \theta_i^{t-1}, W_i^{t-1} - \eta \nabla l(\theta_i^{t-1}, W_i^{t-1}; b)$
14: **end for**
15: Transmit θ_i^t, W_i^t, and n_i to central server
16: **end for**
17: Calculate data samples weight $w_t = \sum_{i=1}^{N_{tr}} n_i$
18: Update model: $\theta_k^t = \sum_{i=1}^{N_{tr}} \frac{n_i * \theta_i^t}{w_t}$, $W_k^t = \sum_{i=1}^{N_{tr}} \frac{n_i W_i^t}{w_t}$
19: The server transmits θ_k^t, W_k^t to k-th contexts
20: **end for**
21: **Output:** θ^T and W^T

A. The Proposed Solution

Our proposed FL solution follows steps as described in algorithm 1. In the first few steps, we initialise the contexts and split them into train and validation sets. After the initialisation stage, the training, locally at each context, take place. At each communication epoch we randomly select N_tr contexts which then train the neural network locally. The results of training, i.e., the trained neural network θ_i and its weights W_i along

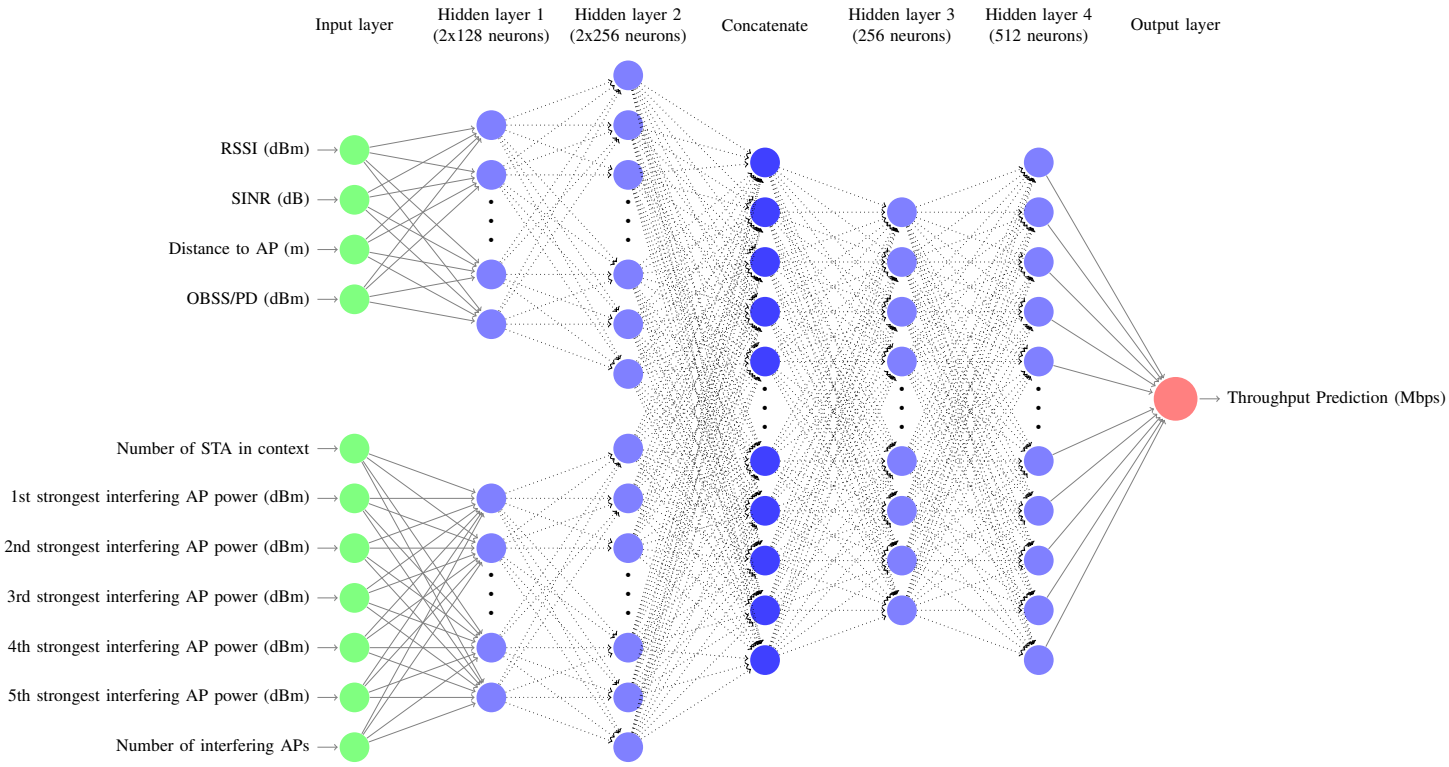

Fig. 4: Visualisation of the DNN structure used.

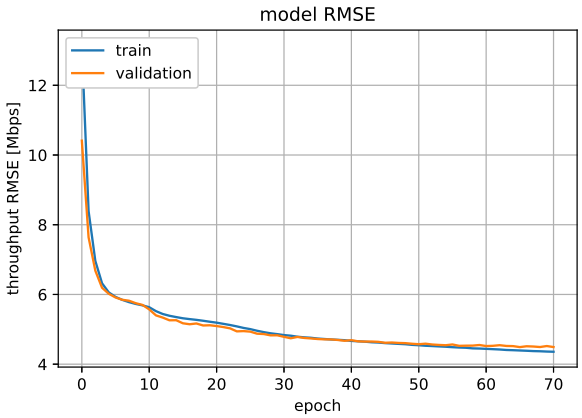

Fig. 5: History of training and validation throughput Root Mean Square Errors (RMSEs).

with the number of data sample n_i, are then transmitted to the central server for merging. After the merger, the central server updates the neural network on every context. The training cycle repeats T-times.

The most important aspect of our solution is combining the trained weights at the central server. Initially weighted the updates from each context equally, but such a an approach resulted in a performance that is skewed toward contexts with more STA. Such behaviour can be attributed to the fact that contexts with four STA have twice as many samples as contexts with only two STA. Therefore, we weight each context update based on the number of data samples belonging to the context during training. This approach deliver the best possible predictions of the throughput.

B. Evaluation

To evaluate the performance of the proposed solution, we relied on two metrics: RMSE and Mean Average Error (MEA). We perform the validation using five per cent of available contexts, randomly sampled from $\mathbb{K}$. During the competition phase, the authors primarily relied on the value RMSE and MEA values of the validation sets when determining which approach was the best fit.

In Fig. 6 we show how the MEA changes over the number of communication rounds. With each communication round, the MEA is decreasing. A trend is similar for contexts we use during the training process (Training set), and for contexts, we use only for validation(Validation set). However, the validation decrease is noisier as it sometimes increases between two consecutive communication rounds. Such behaviour is due to the random sampling approach as not all randomly selected sets $\mathbb{K}_{ep}$ wholesomely represent the system.

Fig. 7 depicts the decrease of RMSE over a number of communication rounds for both training and validation sets. Interestingly, changing the N_{tr} does change the value to which the Throughout RMSE converges, but only impacts the speed, i.e., the number of communication rounds necessary to achieve it. Thus, the larger the N_{tr} is, the faster the system converges.

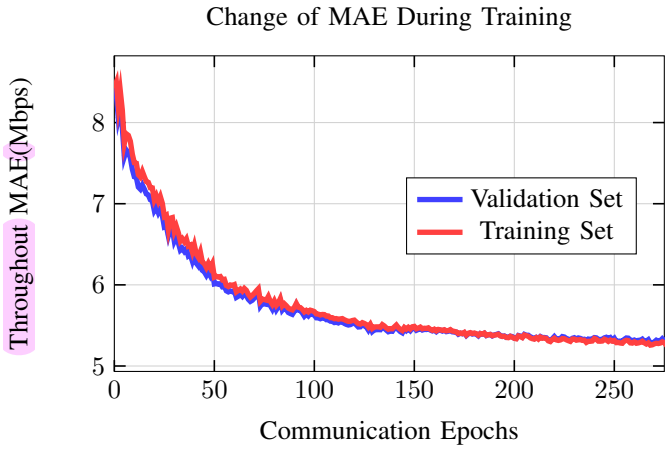

Fig. 6: Change of MEA over number of communication rounds.

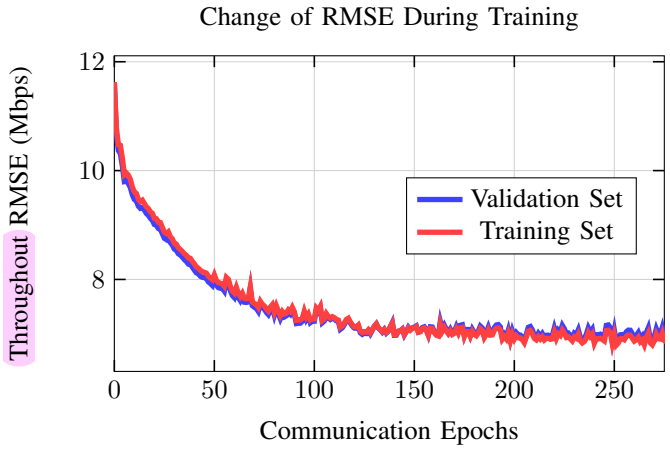

Fig. 7: Change of RMSE over number of communication rounds.

However, the trade-off is the higher overall computational power required to perform the training.

C. Performance on the test dataset

In this section, we discuss the performance of our solution on the test dataset. The accuracy of the submitted predictions was 6.55 Mbps, the second result among all the submitted solutions. In Table I, we report the list of hyper-parameters used in the submitted solution, and which pre-trained DNN can be found in the GitHub repository [].

In Fig. 8, we show the cumulative distribution function (CDF) of true throughout values, i.e., throughout values of the test dataset, predictions we obtain with the idealised central approach, and predictions of our proposed FL approach. The CDF shows the distribution of obtained predictions or true values. In other words, the CDF provides us with insight into which throughput values are more common. For example, more than half of all true throughput values are between 0 and 15 Mbps, i.e., 50-th percentile is 15 Mbps. Interestingly, the obtained distribution closely follows the true values in the idealised central approach (where data points from different contexts are merged). Unfortunately, the FL does not predict any low throughput values, i.e., for throughput values lower

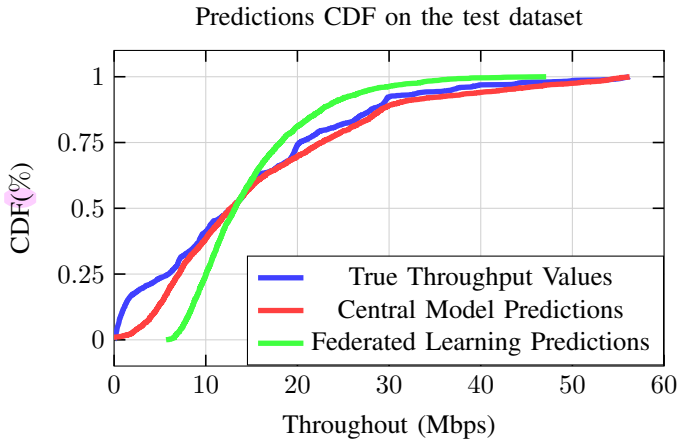

Fig. 8: CDF of predicted throughput we obtained with our federated learning solution compared to a centralised approach and true throughput values.

TABLE I: Hyper-parameters

Federated Learning Parameters	Number of contexts K	1946
	Evaluation contexts N_{eval}	97
	Training contexts N_{tr}	500
	Communication Epochs T	250
Neural Network Training options	Solver name	Adam [7]
	Batch size B	21
	Dropout	10%
	Learning rate	10^{-4}
	L2 regularisation	10^{-5}

than 5 Mbps. We believe that FL approach does not fit well to lowe values due to averaging process of neural networks from different networks and due to the random selection of contexts, which may skip the STA with the lowest throughput. Therefore, our future work will focus on improving the predictions for when the output predictions are low. For example, ensuring that when combining contexts selected for training follow the CDF distribution.

V. Lesson Learned and Conclusion

We tested many different approaches throughout the experimentation process, from altering the methodology of selecting contexts to the neural network design. In this section, we provide our overview of many lessons, i.e., tried approaches that did not perform well, we learned while we were designing a federated learning solution for predicting a throughput in a multi-BSS scenario.

Selecting input features: We considered many different input features as a possible input, but were not employed due to their limited impact on the overall accuracy of predictions. For example, we considered utilising the set of distances from interfering APs to the STA as an input feature. However, using distances along with their power did not improve predictions; thus, we removed them from the final list. Similarly, we noticed a five percent improvement when we normalised the output predictions between 0 and 10, instead of 0 and 1 or 0 and 120.

Neural network design: The selection of the activation function in the neural network has a significant impact on the

achieved accuracy. For example, using ReLU or Leaky ReLU in place of hyperbolic tangent function resulted in around a ten per cent decrease in performance. The performance (measured on the validation set) also deteriorated or remained the same if we increased the number of neurons or added hidden layers to the neural network. Unfortunately, such an approach only resulted in increased training time and required computational power.

Selecting contexts: We tested different strategies for selecting contexts during the training process: grouping by the number of data samples, the number of AP, the number of STA, etc. However, none of the tested strategies resulted in improved performance in comparison to a random selection. For example, selecting only the contexts that have two STA improves RMSE of validation contexts that have two STA. Still, the RMSE decreases when validating with contexts that have three or four STA. Interestingly, the number of selected contexts, i.e., N, impacts only the convergence speed as the greater is the number of contexts used, the faster is the convergence.

The proposed solution performed reasonably well as it achieved 6.55 Mbps MEA on the tested dataset and was ranked second in terms of achieved throughout prediction accuracy. We believe the base of our solution, combined with insights gained by others teams, can pave the way to create a solution capable of selecting the best OBSS/PD threshold value for a given system.

REFERENCES

[1] F. Wilhelmi, S. Barrachina-Muñoz, C. Cano, I. Selinis, and B. Bellalta, "Spatial reuse in IEEE 802.11ax wlans," *arXiv e-prints*, vol. abs/1907.04141, 2019.

[2] I. Selinis, K. Katsaros, S. Vahid, and R. Tafazolli, "Control OBSS/PD sensitivity threshold for IEEE 802.11ax BSS color," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1–7.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] S. Barrachina-Muñoz, F. Wilhelmi, I. Selinis, and B. Bellalta, "Komondor: a wireless network simulator for next-generation high-density wlans," *arXiv e-prints*, vol. abs/1811.12397, 2018.

[5] F. Wilhelmi, S. Barrachina-Muñoz, and B. Bellalta, "On the performance of the spatial reuse operation in IEEE 802.11ax WLANs," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2019, pp. 1–6.

[6] The TensorFlow Federated Authors, "Tensorflow federated," GitHub repository: https://github.com/tensorflow/federated", 2018.

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.