# 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*—Due to the distributed channel access policies combined with the usage of unlicensed frequency bands, Wi-Fi system performance tends to degrade in crowded scenarios like airports, stadiums and public gatherings. To increase simultaneous transmissions and improve Wi-Fi performance, the recent IEEE 802.11ax amendment includes spatial reuse (SR), which is expected to continue evolving in IEEE 802.11be and future amendments. However, one of the main limitations of the SR mechanism is that it relies on local information, limiting the effectiveness of SR techniques. This report proposes a distributed method based on Federated Learning (FL), which enables the selection of the best configuration of Preamble Detection (PD) threshold for Overlapping Basic Service Set (OBSS) by predicting the achievable throughput without centralised training data. 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 Federated Learning (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 Overlapping Basic Service Set (OBSS)-Preamble Detection (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 for training to transfer from the edge to a central server and stringent General Data Protection Regulation (GDPR) policies for data privacy make centralised ML techniques challenging to be applied in practice. Moreover, in a multi-operator setting, sharing data between BSS of different vendors may reveal sensitive information and disclose proprietary design and vendor-specific implementations. Instead, the FL approach performs the training on the edge device. Only the result of the training, namely the model parameters, are exchanged with the central server.

Fig. 1 shows the system model, which consists of $K$ contexts and a central server. We define a set $\mathbb{K} = \{k_1, k_2, \dots, k_K\}$ 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 proprieties such as AP and STA devices locations, AP load and interfering conditions. In each context a reference BSS tests different OBSS-PD configurations in the range $[-82, -62]$ 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 converges.

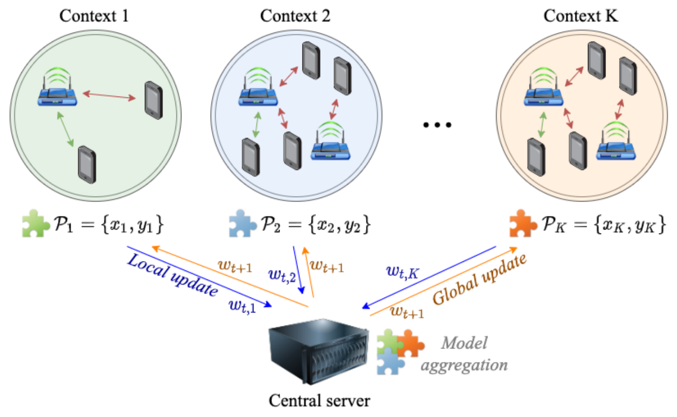

Fig. 1: Representation of the FL system with different contexts formed by different Access Points (APs) and STAs. Each context trains the ML model with local data and sends the update to a central server that communicates the global update after model aggregation.

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. VI, we

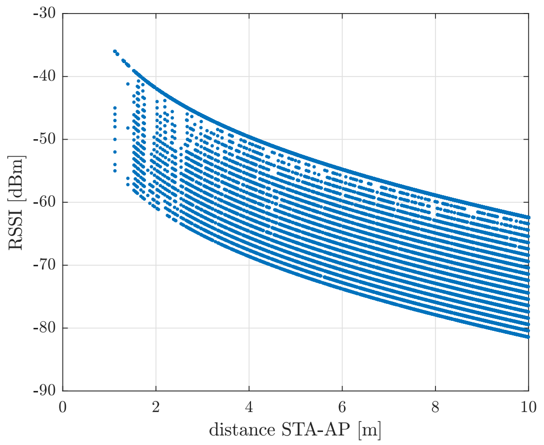

Fig. 2: Representation of the attenuation of the Received Signal Strength Indicator (RSSI) increasing the distance between STA and AP for each configuration of OBSS-PD threshold in the range $[-82, -62]$ dBm and represented in ascending order from top to bottom.

document the lessons learned from this experience.

## II. Data Analysis

In this part, we discuss how features are selected for the training process. We use Version 1.3 of the dataset [4] created for the problem statement ITU-ML5G-PS-004 of the ITU AI/ML Challenge (2021 edition). More specifically, we consider Scenarios 2 and 3, containing 2000 different IEEE 802.11ax deployments with 2-6 APs and 1-4 STAs per AP. We start extracting the set of features available in the simulator's output file of each context, namely the OBSS-PD configuration, the 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 the 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 these information can also be obtained online in a real system. Indeed, 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, employing multi-antenna processing techniques to separate the different interfering sources. In addition, Time-of-Arrival (TOA) ranging techniques can determine the distance between STA and AP .

Before obtaining the final dataset, we preprocess the data through different steps. First, we clean the input and output data parsed from the simulator files 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 minimum recorded according to our system settings, 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 in the range $[-82, -62]$ dBm and represented 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 [5]. 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 [6]. Therefore, as a design choice in the data preprocessing, 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 preprocessing 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 strong 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 characterise with domain-based 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

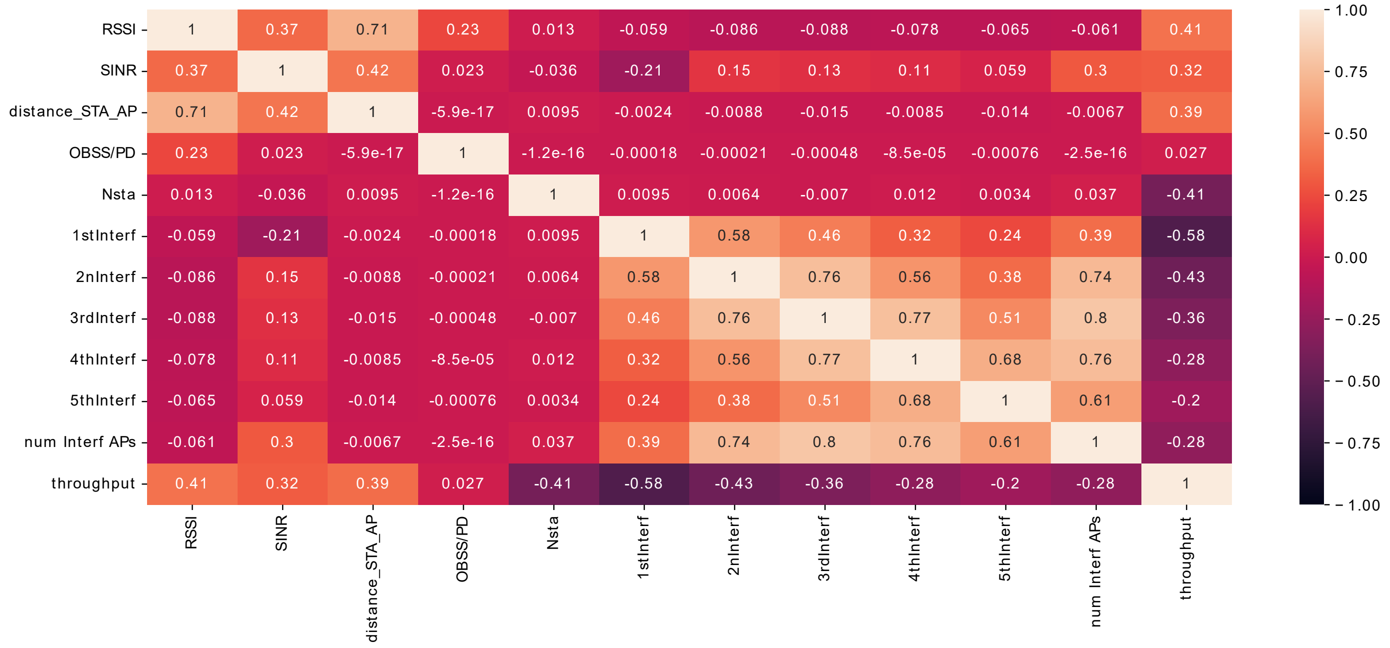

Fig. 3: Representation of correlation matrix showing the correlation between different input and output variables of the dataset.

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.

## 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 power received from interfering APs 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 hyperbolic tangent (*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 Root Mean Square Error (RMSE) obtained with centralised training. One DNN model is used, and it takes as input the data from all the contexts. This represents the lower bound achievable by the FL solution and guides 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 solution is based on the implementation outlined in [7]. In addition, our approach combines the trained weights in a novel way, which is designed specifically for this challenge and is based on the data acquired in each context.

### A. The Proposed Solution

Our proposed FL solution follows the steps described in Algorithm 1. In steps 1-5, we initialize the contexts and split them into train and validation sets. After the initialization stage, the training takes place locally in a subset of $N_{tr}$ contexts, which are selected randomly at every communication epoch. The training results, i.e., the trained neural network $\theta_i$ and its weights $w_i$ along with the number of data sample $n_i$, are then transmitted from the $N_{tr}$ contexts to the central server for aggregation. After the aggregation, the central server sends back the global trained model to every context, which updates their local DNN models. The training cycle repeats for $T$ communication epochs.

One of the most important aspects of the FL training consists of aggregating the weights at the central server. Initially, we weighted the updates from each context equally, but such an approach resulted in a skewed performance 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. Instead, our solution proposed to weight each context update based on the number of data samples used for the training in each context normalised by the number of STA in the contexts. We denote this normalisation

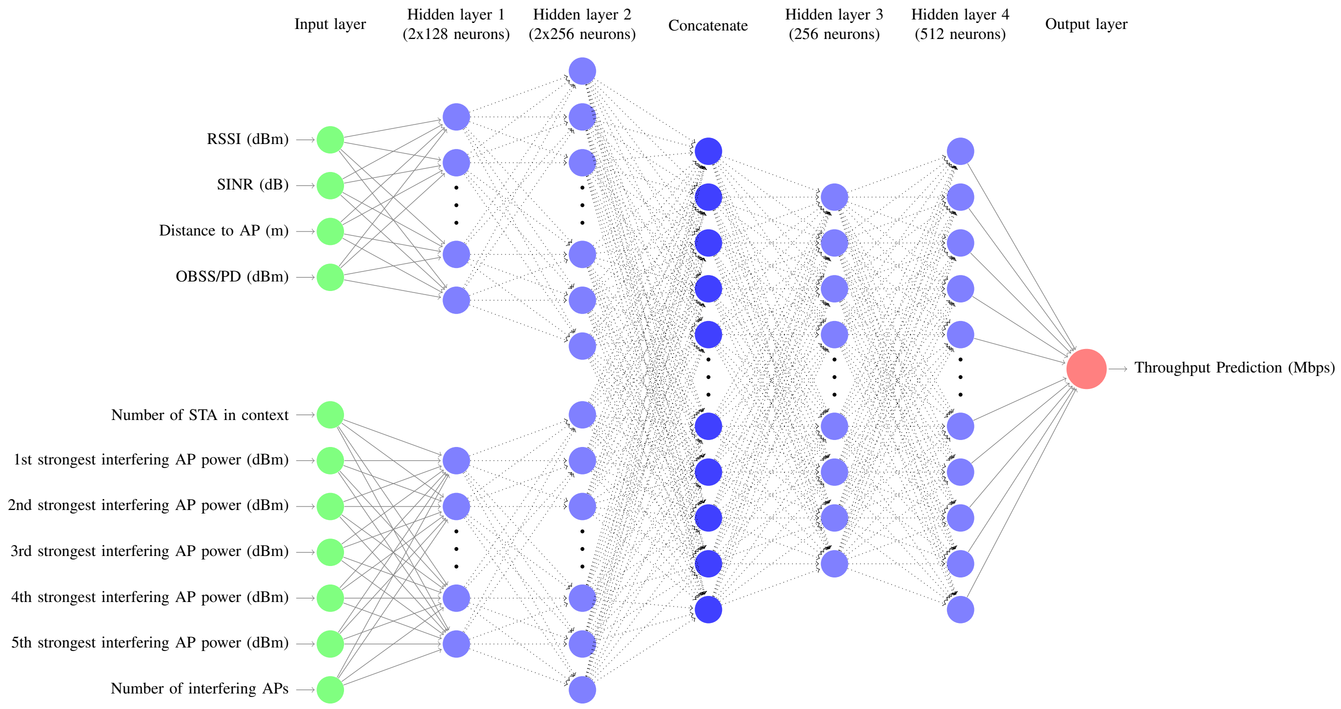
Input layer
Hidden layer 1 (2x128 neurons)
Hidden layer 2 (2x256 neurons)
Concatenate
Hidden layer 3 (256 neurons)
Hidden layer 4 (512 neurons)
Output layer
RSSI (dBm)
SINR (dB)
Distance to AP (m)
OBSS/PD (dBm)
Number of STA in context
1st strongest interfering AP power (dBm)
2nd strongest interfering AP power (dBm)
3rd strongest interfering AP power (dBm)
4th strongest interfering AP power (dBm)
5th strongest interfering AP power (dBm)
Number of interfering APs
Throughput Prediction (Mbps)

Fig. 4: Visualisation of the DNN structure used.

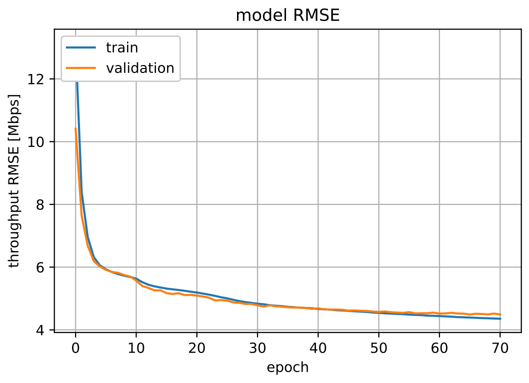

Fig. 5: History of training and validation throughput RMSE obtained with a centralised training.

weights with $\alpha$. This approach improves the accuracy of the predictions of the throughput.

## B. Evaluation

To evaluate the performance of the proposed solution, we consider both RMSE and Mean Average Error (MAE) metrics. We perform the validation using five per cent of available contexts, randomly sampled from $\mathbb{K}$. In total we had 1946 available contexts, i.e., $K = 1946$. Five percent of available contexts were used for evaluation, i.e., $N_{eval} = 97$. At every communication round, we selected 500 contexts randomly to perform training on, i.e., $N_{tr} = 500$. Furthermore, the contexts we use during the training and validation set are kept separated to prevent the data leakage and to be able to recognise when the solution starts to over-fitting to the training data.

**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 $\theta^0$ and $W^0$
5: The server transmits $\theta_k^0, w_k^0$ 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:         Split context's samples in $\beta$ ($\frac{n_i}{B}$ batches of size $B$)
10:        Where $n_i$ represents the number of data samples
11:        for batch $b$ in $\beta$ do
12:            $\theta_i^t, w_i^t \leftarrow \theta_i^{t-1}, w_i^{t-1} - \eta \nabla l(\theta_i^{t-1}, w_i^{t-1}; b)$
13:        end for
14:        Determine weight $\alpha_i = n_i / N_{STA}$
15:        Transmit $\theta_i^t$, $w_i^t$, and $\alpha_i$ to central server
16:    end for
17:    Calculate data samples weight $\alpha_t = \sum_{i=1}^{N_{tr}} \alpha_i$
18:    Update model: $\theta_k^t = \sum_{i=1}^{N_{tr}} \frac{\alpha_i * \theta_i^t}{\alpha_t}$, $w_k^t = \sum_{i=1}^{N_{tr}} \frac{\alpha_i w_i^t}{\alpha_t}$
19:    The server transmits $\theta_k^t, w_k^t$ to $k$-th contexts
20: end for
21: Output: $\theta^T$ and $w^T$
```

In Fig. 6 we show how the MAE decreases over the number of communication epochs. The same trend occurs for the contexts that we use during the training process (Training set) as well as for the contexts that we use only for validation

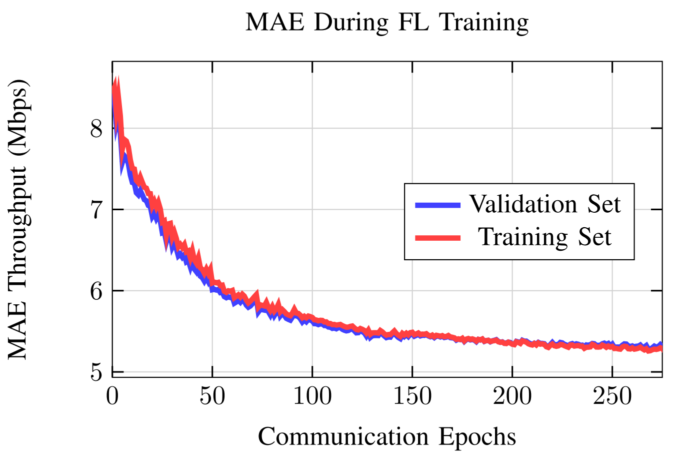

Fig. 6: Representation of MAE over the number of communication epochs.

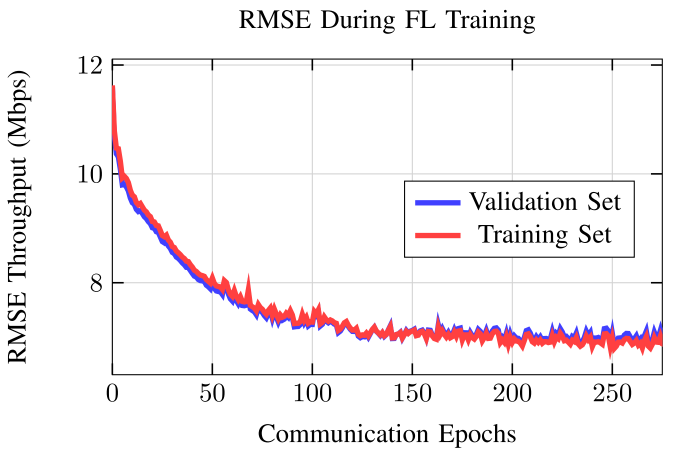

Fig. 7: Representation of RMSE over the number of communication epochs.

(Validation set). However, the validation throughput decrease is noisier as it sometimes increases between two consecutive communication epochs. 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 epochs for both training and validation sets. Interestingly, changing the $N_{tr}$ does impact the value achieved by the RMSE but only affects the convergence time, i.e., the number of communication epochs necessary to achieve it. Thus, the larger the $N_{tr}$ is, the faster the system converges. However, the overall computational resources required increase considerably as more contexts are trained in parallel. Hence there is a trade-off between the training delay and the number of contexts selected 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, which is the second result among all the solutions submitted. We used a neural network that was trained for 250 communication rounds, i.e., $T = 250$. In Table I,

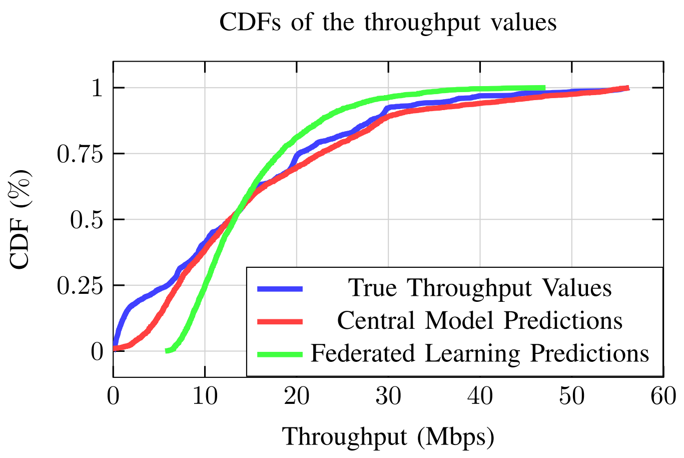

Fig. 8: CDF of predicted throughput we obtained with our federated learning solution compared to a centralized approach and true throughput values.

TABLE I: Hyper-parameters

| | | |
|---|---|---|
| Neural Network Training options | Solver name | Adam [9] |
| | Batch size $B$ | 21 |
| | Dropout | 10% |
| | Learning rate | $10^{-4}$ |
| | L2 regularisation | $10^{-5}$ |

we report the list of hyper-parameters used in the submitted solution and related pre-trained DNN can be found in the GitHub repository [8].

In Fig. 8, we show the cumulative distribution function (CDF) of actual throughput values, the CDF of throughput predicted with DNN model with centralised training and the CDF of throughput predicted with our proposed FL distributed approach. The CDF provides us with insights about the statistical distribution of the throughput values for all the STAs of the network. At the 50-th percentile of the CDF (median), the predicted throughput is 15 Mbps and matches both the actual values and the one of the centralized training, which we recall, performs the training aggregating the data from all the contexts. On the other hand, the FL does not predict well both low and high throughput values. For these values, the centralised model show a better fit to the actual throughput values. We believe that the FL approach does not fit well due to the averaging process implemented while aggregating the updates of the model from different contexts. Moreover, the random selection of contexts may skip the STA with the lowest throughput. Therefore, our future work will focus on improving the predictions focusing on the lower tail of the CDF. For example, ensuring that when combining contexts selected for training these well represent the CDF distribution.

## V. Lesson Learned and Conclusion

In this section, we provide our overview of many approaches that we tried but did not perform well. We learned many lessons while designing the FL solution as we tested many different directions throughout the experiments, and we want to summarise as follows:

**Selecting input features:** We considered many different input features as a possible input, but not all of them were employed due to their limited impact on the overall accuracy of predictions. For example, we considered utilising as additional input features the set of distances from interfering APs to the STA. However, using these distances together with the received power from interfering APs did not improve predictions accuracy. Thus, we removed them from the final list. Similarly, we noticed a five per cent 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. Such approach was discarded as it only resulted in increased training time and required computational power.

**Selecting contexts**: We tested different strategies for selecting contexts during the training process: grouping contexts by the number of data samples, grouping contexts by the number of AP, and grouping contexts by the number of STA. However, none of the strategies resulted in an 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., $N_{tr}$, impacts only the convergence speed as the greater is the number of contexts used, the faster is the convergence time.

The proposed solution performed reasonably well as it achieved MAE of $6.55$ Mbps on the test 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.

## VI. Acknowledgements

We would like to thank Dr Francesc Wilhelmi for his comments and for organising the challenge related to the problem statement ITU-ML5G-PS-004 of the ITU AI/ML Challenge (2021 edition).

## 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. Konečný, 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] F. Wilhelmi, "ITU AI/ML Challenge 2021 Dataset IEEE 802.11ax Spatial Reuse," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5506248

[5] 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.

[6] 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.

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

[8] J. Hribar and A. Bonfante, "FederationS: Federated Learning for Spatial Reuse in a multi-BSS (Basic Service Set)scenario," Nov 2021. [Online]. Available: https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-004_Federated_Learning_team_FederarationS

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