FEDERATED SPATIAL REUSE OPTIMIZATION IN NEXT-GENERATION DECENTRALIZED IEEE 802.11 WLANS

Francesc Wilhelmi[1], Jernej Hribar[2], Selim F. Yilmaz[3], Emre Ozfatura[3], Kerem Ozfatura[3], Ozlem Yildiz[4], Deniz Gündüz[3], Hao Chen[5], Xiaoying Ye[5], Lizhao You[5], Yulin Shao[5], Paolo Dini[1], Boris Bellalta[6]
[1]CTTC (Spain), [2]CONNECT Centre, Trinity College Dublin (Ireland), [3]Imperial College London (United Kingdom), [4]New York University (USA), [5]Xiamen University (China), [6]Universitat Pompeu Fabra (Spain)

NOTE: Corresponding author: Francesc Wilhelmi, francesc.wilhelmi@cttc.cat

Abstract *– As wireless standards evolve, more complex functionalities are introduced to address the increasing requirements in terms of throughput, latency, security, and efficiency. To unleash the potential of such new features, artificial intelligence (AI) and machine learning (ML) are currently being exploited for deriving models and protocols from data, rather than by hand-programming. In this paper, we explore the feasibility of applying ML in next-generation wireless local area networks (WLANs). More specifically, we focus on the IEEE 802.11ax spatial reuse (SR) problem and predict its performance through federated learning (FL) models. The set of FL solutions overviewed in this work is part of the 2021 International Telecommunication Union (ITU) AI for 5G Challenge.*

Keywords – Federated learning, IEEE 802.11ax, ITU Challenge 2021, machine learning, network simulator, spatial reuse

1. INTRODUCTION

Wireless networks are evolving towards artificial intelligence (AI) / machine learning (ML)-driven systems able to address the overwhelming requirements of future mobile communications [1, 2], namely the fifth generation (5G), beyond 5G (B5G), and the sixth generation (6G). The application of ML for networking can be found at different communication layers and parts of a network, e.g., network management to drive the self-organizing networks (SON) paradigm [3], optimization of the medium access control (MAC) layer in decentralized channel access [4], or AI-native physical communication protocols [5, 6]. The fact is that AI/ML can leverage the vast amount of network and user data to generate new knowledge that allows improving the network performance and, hence, making progress in the development of novel network applications such as those based on extended reality.

Nevertheless, the use of ML in communications also raises concerns of different nature. First, ML-based solutions typically require a lot of energy for training complex models (e.g., neural networks) and high bandwidth for exchanging training data, which typically needs to be centralized to a single point. Moreover, the massive usage of networking data for ML may threaten security and users' privacy. The privacy issue may be exacerbated in decentralized networks such as IEEE 802.11 wireless local area networks (WLANs), whereby the lack of a central network manager may make inter-WLAN interactions unreliable.

To address some of the challenges posed by traditional ML training, Federated Learning (FL) optimization was introduced in [7] as a distributed training paradigm that allows keeping the data at its source. Since then, a significant number of FL applications have flourished across different fields, such as medicine [8], autonomous driving [9], UAV-based wireless networks [10]. FL has become attractive to foster collaboration among different parties interested in solving a common problem. Under the management of a central server (typically, a neutral entity), FL participants contribute to building a common ML model, by sharing model weights generated using its own local data, rather than forwarding raw data for centralized training.

In this paper, we study the application of FL models to the IEEE 802.11 spatial reuse (SR) problem, which aims to enhance spectral efficiency by adjusting the devices' carrier sense area to increase the number of concurrent transmissions in overlapping deployments. IEEE WLANs are an important part of the B5G ecosystem as it represents a cost-effective but high-performance solution for the access network. In particular, we overview the output of the problem statement entitled *"ITU-ML5G-PS-004: Federated Learning for Spatial Reuse in a multi-BSS (Basic Service Set) scenario"*, which was part of the 2021 International Telecommunication Union (ITU) AI for 5G Challenge [11].[1] In the proposed problem statement, a dataset with simulated IEEE 802.11ax (11ax) measurements was provided to develop FL solutions that are able to predict and optimize the perfor-

[1]The ITU AI/ML challenge is a global competition that gathers professionals, researchers, practitioners, and students from all around the globe to solve relevant problems on ML for communications.

Table 1 – Summary of the ML models proposed by the participants of the challenge.

Team	Proposed Model	Motivation	Ref.
FederationS	DNN with two parallel branches	Exploit relationships among training features	[14]
FedIPC	NN with a Multi-Output Regression Objective	Take advantage of knowledge on wireless operation	[15]
WirelessAI	CNN with FCNN	Exploit graph representations in wireless networks	[16]

mance of next-generation crowded Wi-Fi deployments applying SR. The usage of simulated data for enriching training datasets is another relevant topic for enabling ML in communications [12].

The main contributions of this paper are as follows:

- We overview the SR technology for both 11ax and future amendments and propose the usage of FL to address it.
- We provide a dataset with 11ax SR measurements for next-generation WLANs. The dataset is open and can be accessed at [13].
- We overview the set of FL solutions proposed by the participants of the 2021 ITU AI for 5G Challenge. Table 1 briefly summarizes the proposed models, as well as the main motivation behind them.

The rest of the paper is structured as follows. Section 2 introduces the SR problem in 11ax and future WLANs. Section 3 provides some basics on FL with special emphasis on its applications in networking. Section 4 overviews the provided SR dataset for training ML models. The solutions proposed by the challenge participants are described in detail in Section 5 and evaluated in Section 6. Section 7 concludes the paper with some remarks and future directions.

2. SPATIAL REUSE IN 802.11AX WLANS: OVERVIEW AND RESEARCH GAPS

IEEE 802.11 technology, commonly known as Wi-Fi, is one of the most popular solutions for the access network due to its ease of deployment and low cost (it operates on unlicensed bands). However, its fundamental operation is based on carrier sense multiple access (CSMA), whose performance is well known to degrade when dealing with a large number of concurrent users [17]. To address the issues raised by network density and to meet the increasingly strict requirements posed by next-generation applications (e.g., virtual reality), 802.11 amendments introduce novel functionalities and protocol enhancements. For instance, standards 802.11n (2009) and 802.11ac (2013) provided high throughput (HT) and very high throughput (VHT) devices by including, for instance, the application channel bonding (CB), whereby basic channels could be aggregated to increase the capacity of a single transmission.

As for the SR operation [18], it was recently introduced by the IEEE 802.11ax (2021) standard [19] to increase the number of parallel transmissions in overlapping basic service sets (OBSS). Among other features like orthogonal frequency division multiple access (OFDMA), or downlink/uplink multi-user multiple input-multiple-output (MU-MIMO), SR aims at enhancing the performance and efficiency. To do so, it provides two different operational modes:

1. OBSS Packet Detect-based SR (OBSS/PD-based SR).
2. Parametrized Spatial Reuse (PSR).

The main difference between the two mechanisms lies in the way SR transmission opportunities (TXOPs) are detected by devices implementing them. While OBSS/PD-based SR operates in the downlink, PSR is designed for the uplink. In what follows, we focus on OBSS/PD, which has gained more interest and is under consideration for evolution in the future amendments, such as the IEEE 802.11be (11be) [20]. A comprehensive overview of these two mechanisms can be found in [18].

In essence, OBSS/PD-based SR allows devices to transmit in parallel with others that gained channel access beforehand. To do so, a new OBSS/PD threshold is defined to be applied when an incoming detected transmission marks the radio channel as busy through clear channel assessment (CCA) operation. CCA allows overlapping devices to share a common channel and is triggered when the preamble of a Wi-Fi transmission is identified. Provided that the OBSS/PD threshold allows initiating a new SR transmission, a transmit power limitation must be applied so that the generated interference does not affect the original transmission.

The OBSS/PD SR operation is illustrated in Fig. 1 for two overlapping access points (APs), AP_A and AP_B. As shown in Fig. 1a, AP_A detects the signals from all the considered devices (represented by the red area), including AP_B and station B (STA_B), which belong to a different BSS. In particular, AP_B is inside the carrier sense area of AP_B (represented by the gray area), so they both must contend for the channel whenever the other starts a transmission (e.g., to its associated STAs). Nevertheless, thanks to the OBSS/PD-based SR operation, AP_A can ignore AP_B's transmissions when applying the OBSS/PD threshold (represented by the green area). At the packet level (shown in Fig. 1b), AP_A starts decoding the preamble of a new transmission from AP_B, which has initially gained the access to the medium using CSMA with collision avoidance (CSMA/CA). From the preamble reception, AP_A determines that the channel is busy at the MAC layer due to the CCA operation. But, using the SR mechanism, AP_A identifies an SR TXOP because the incoming signal is below the OBSS/PD threshold. Hence, AP_A can

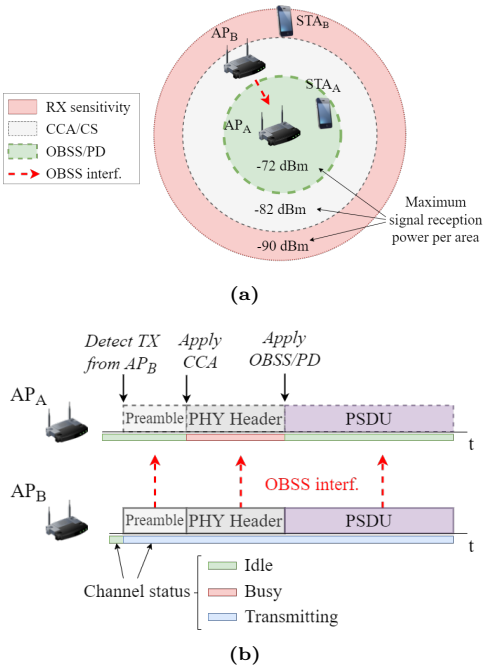


Fig. 1 – IEEE 802.11ax OBSS/PD-based SR operation: (a) signal reception areas, (b) diagram of packet exchange.

initiate a transmission before AP_B leaves the channel, provided that a transmit power restriction is applied, denoted by TX_PWR_{max}, for the sake of not affecting AP_B's transmission:

$$TX\_PWR_{max} = TX\_PWR_{ref} - (OBSS/PD - OBSS/PD_{min}), \quad (1)$$

where TX_PWR_{ref} is the transmit power reference (set to 21 dBm or 25 dBm, depending on the device's antenna capabilities), OBSS/PD is the selected OBSS/PD threshold for detecting SR TXOPs, and $OBSS/PD_{min}$ is the minimum OBSS/PD threshold (fixed to -82 dBm).

While SR promises to enhance spectral efficiency in dense OBSS deployments, its actual performance is hindered by the proper selection of the OBSS/PD threshold, which may not be trivial due to the complex inter-device interactions in a WLAN. The fact is that the OBSS/PD-based SR operation is a decentralized mechanism that only considers the interactions between principal transmitters (i.e., devices gaining access to the channel for transmitting), but does not account for either the interference at the recipients of such transmissions or the impact of uplink control frames (e.g., acknowledgment packets). Since the standard does not provide any method for selecting the proper OBSS/PD threshold, there is an imperative need for finding effective mechanisms to leverage the SR operation.

ML, in this context, is considered as a promising tool to capture the complex interactions among IEEE 802.11 devices applying SR. Most of the literature in this subject has so far focused on reinforcement learning (RL) and online learning techniques, whereby agents attempt to learn the best OBSS/PD configuration sequentially. In [21, 22], the authors modeled the decentralized SR problem as multi-armed bandits (MAB), an online learning framework whereby agents attempt to address the exploration-exploitation trade-off. While [21] studied the problem by using selfish rewards in a competitive environment, [22] considered shared rewards for the sake of maximizing fairness. Other RL-based approaches can be found in [23] and [24].

The online learning paradigm turns out to be a cost-effective solution to the decentralized SR problem thanks to its ability for solving complex partial information problems. In addition, WLANs typically experience a high variability both in terms of devices' mobility and activation/deactivation, so past learned information may become easily outdated. However, as shown in [22], online learning may have some pitfalls when applied to dense WLANs, mainly raised by the high action-decision space, the non-stationarity of agents' rewards in competitive settings, or the complexity of finding a proper shared reward that enables maximizing the overall network performance.

For those reasons, in this paper, we focus on the suitability of supervised learning methods, mostly based on deep learning (DL), for the SR problem in WLANs. To the best of our knowledge, this approach has not been studied before in the context of SR. A centralized DL-based method was proposed in [25] to jointly select the transmission power and the CCA, but not in the context of 11ax SR operation. DL was also applied in [26] to address the channel bonding problem in dense WLANs. This and other DL solutions for the dynamic channel bonding problem in IEEE 802.11ax WLANs were overviewed in [27]. For further details on Wi-Fi optimization through ML, we refer the interested reader to the comprehensive survey in [28].

We note that the overviewed works on DL consider centralized approaches, which require data to be gathered at a single point for training a static model, which is then used homogeneously across all the AI-enabled devices. Nevertheless, in practice, some deployments (e.g., residential WLANs) may have limitations in terms of computation, storage, or communication capabilities (for instance, low-throughput connections, intermittent availability). Moreover, separate WLAN deployments can be substantially different, thus requiring specialized models (rather than general ones). To address these limitations of centralized learning on heterogeneous deployments, we focus on the FL paradigm, introduced in the next section.

3. AN INTRODUCTION TO FEDERATED LEARNING FOR NETWORKING

The FL optimization paradigm was first introduced in [29] to address some critical issues of traditional cen-

tralized ML mechanisms. In FL, training is done at end devices (or clients), which do not share their training data with others. Instead, ML model updates are provided and aggregated under the management of a typically central server. By removing the procedures related to data exchange, FL decreases the communication overhead and enhances user privacy and security. Moreover, FL is an appealing solution for dealing with heterogeneous sets of clients, thus allowing to create specialized models according to clients' characteristics. With that, FL has the potential to revolutionize ML implementations, bringing them closer to practical applications and use cases. Many examples of FL have emerged in recent years, including, but not limited to, in medicine [30], finance [31], industry 4.0 [32], or telecommunications [33, 34].

In the telecommunications realm, novel ML solutions require handling a vast amount of data, often highly distributed across the network. These kinds of resource-demanding applications may threaten the stability of the network on the one hand and may experience low performance due to the communication bottleneck on the other hand. FL can potentially alleviate some of these issues by reducing the overheads generated by the ML operation while providing good performance. FL also contributes to enhancing privacy, which is a critical issue in communications. FL applications in communications [35] include autonomous driving [36], unmanned aerial vehicle (UAV)-based wireless networks [37], edge computing [38], physical layer optimization [39], or Internet-of-Things (IoT) intelligence [40].

The generic FL algorithm operates iteratively, generating a global model update at each iteration with the help of a subset of clients. Each algorithm's iteration follows the following general steps (see Fig. 2):

1. A set of $\mathcal{K} = \{1, 2, ..., K\}$ clients download the current model parameters, w_t, from the central server (also called the parameter server).

2. Clients perform training in parallel using their local datasets $\mathcal{D}^{(k)}$ (with size $N^{(k)}$) and update the model weights accordingly, denoted by $w^{(k)} \in \mathbb{R}^d$.

3. The server pulls the model updates from the participating clients (a subset of clients may be selected in each FL round for the sake of performance) and orchestrates weight aggregation to generate an updated global model $w_{t+1} \in \mathbb{R}^d$.

4. Above steps are repeated until convergence, i.e., until a time horizon is completed or a certain accuracy goal is met.

At this point, it is important to highlight the federated averaging (FedAvg) method [41], which is based on stochastic gradient descent (SGD) optimization and performs well for non-convex problems. In FedAvg (shown in Alg. 1), clients perform several batch updates at each iteration using local data to update the global model parameters. Unlike in classical federated stochastic gradient descent (FedSGD), where gradients are exchanged, FedAvg considers sharing model updates (e.g., the parameters of a neural network). By applying multiple rounds of training, FL seeks to minimize a global finite-sum cost function $l(w)$ by optimizing the global model parameters w:

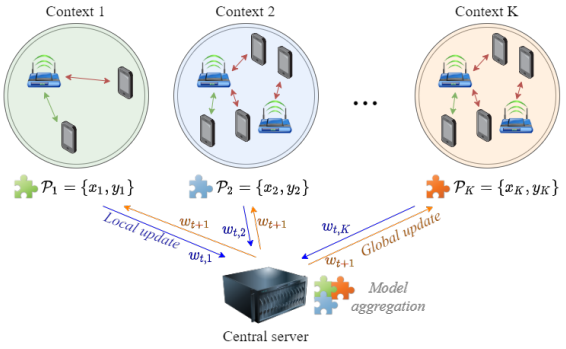


Fig. 2 – FL optimization in different WLAN contexts.

$$\min_{w \in \mathbb{R}^d} l(w) = \min_{w \in \mathbb{R}^d} \sum_{k=1}^{K} \frac{N^{(k)}}{N} l^{(k)}(w, \mathcal{D}^{(k)}), \qquad (2)$$

where $l^{(k)}(w, \mathcal{D}^{(k)})$ is the loss experienced by client k when using the global model w on its local data, and N is the total size of the distributed dataset, i.e., $N = \sum_{\forall k \in \mathcal{K}} N^{(k)}$.

To compute local updates, clients run E epochs of SGD based on the target local loss function $l^{(k)}$ and the batch size B applied to local data $\mathcal{D}^{(k)}$. Using a learning rate η, local updates are obtained by:

$$w_{t+1}^{(k)} \leftarrow w_t^{(k)} - \eta \nabla l^{(k)}(w_t, \mathcal{D}^{(k)}) \qquad (3)$$

Finally, being η the learning rate, the server aggregates clients' weights based on the importance α_k assigned to each client which may be set according to local dataset lengths (as indicated in Eq. *(2)*):

$$w_{t+1} = \sum_{k=1}^{K} \alpha^{(k)} w_{t+1}^{(k)} \qquad (4)$$

Beyond FedAvg, other optimization mechanisms have been proposed to improve the convergence and efficiency of FL [42, 43]. For further details on FL, we refer the interested reader to the works in [44, 45], and to [46] for the implementations of FL over wireless networks in particular.

4. OPEN SIMULATED DATASET ON IEEE 802.11AX SR

Supervised ML methods typically require a significant amount of high-quality data to perform well. Training

Algorithm 1 Federated Averaging (FedAvg)

for $t = 1, 2, \ldots, T$ **do**
  for $k \in \mathbb{K}_{tr}$ **do** in parallel
    Pull $\boldsymbol{w}_t$ from central server: $\boldsymbol{w}_{t,0}^{(k)} = \boldsymbol{w}_t$
    for $e = 1, \ldots, E$ **do**
      Update model: $\boldsymbol{w}_{t,e}^{(k)} = \boldsymbol{w}_{t,e}^{(k)} - \eta_t \nabla l_{t,e}^{(k)}$
    end for
    Push $\boldsymbol{w}_{t+1}^{(k)} \leftarrow \boldsymbol{w}_{t,E}^{(k)}$
  end for
  FedAvg: $\boldsymbol{w}_{t+1} = \frac{1}{|\mathbb{K}_{tr}|} \sum_{k \in \mathbb{K}_{tr}} \boldsymbol{w}_{t+1}^{(k)}$
end for

data is usually obtained either from network activity [47] or from measurement campaigns [48]. However, obtaining real traces from networks can be challenging due to proprietary limitations (data owners are reluctant to share their assets), data privacy issues (most network data is generated by final users), or difficulties in obtaining data from a rich set of situations (anomalies are hard to reproduce and identify). In this sense, the usage of synthetic datasets for model training is gaining attention [12]. Such datasets can be obtained, for instance, from network simulators (e.g., ns-3, OMNET++, OPNET). Simulators are a cost-effective solution for generating comprehensive datasets. Some prominent examples of synthetic datasets oriented to ML training can be found in [49, 50].

As for the provided dataset on 11ax SR [13], it has been generated with Komondor [51], an open-source IEEE 802.11ax-oriented simulator that includes features like channel bonding or SR. Komondor does not implement the targeted functionalities, but its execution is also lightweight, thus allowing for generating large datasets corresponding to massive WLAN deployments.

The dataset contains both training and test files, which include the results obtained from several simulated random deployments applying 11ax SR (see the example random deployment in Fig. 3). More specifically, a set of three baseline scenarios was considered to represent different types of deployments. Considering the features in each type of scenario (e.g., maximum number of STAs per BSS, minimum distance between APs), 1,000 random deployments of each type were generated for training. Each simulated deployment corresponds to a context $k \in \mathbb{K}$, where the BSS of interest (namely, BSS_A) is used as a client for FL optimization. To enrich contexts with data, each BSS includes information for each possible OBSS/PD configuration τ (i.e., from -82 dBm to -62 dBm with 1 dBm precision). Finally, for the test dataset, 1,000 more deployments were simulated using more relaxed constraints. In this case, a single random OBSS/PD configuration was selected in each deployment. Table 2 provides an overview of the entire dataset.

Training scenario *training1* considers BSSs with only one

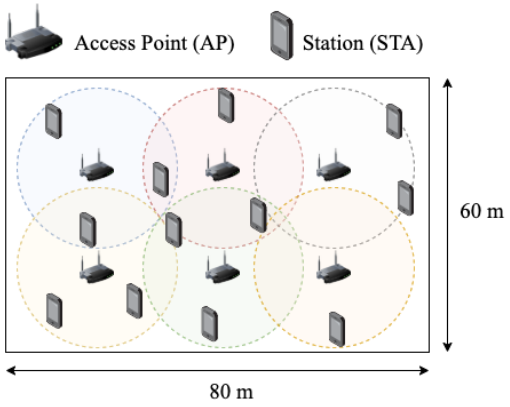


Fig. 3 – Example of a simulated WLAN deployment.

Table 2 – Summary of the scenarios of the dataset.

	Sce id	Num. APs	Num. STAs	d_min(APs)	Context variations
Training	training1	2-6	1	10 m	None
	training2		1-4	10 m	None
	training3		1-4	None	Up to 20 locations
Test	test		2-4	None	None

STA, which is useful to minimize the impact of uplink transmissions, thus allowing to focus on inter-AP interactions only. In contrast, scenarios *training2* and *training3* consider up to 4 STAs per AP, which contribute generating more traffic in the uplink. As for the minimum distance between APs (d_min), it is set to 10 m in scenarios *training1* and *training2*, whereas the rest have no limitation. Furthermore, contexts in scenario *training3* contain richer datasets by simulating variations of the same deployments using different STA locations.

The information included in simulated files is divided into features and label. Concerning features for training, we find the following key elements:

1. **Type of node:** indicates whether the node is an AP or an STA.
2. **BSS id:** identifier of the BSS to which the node belongs.
3. **Node location:** {x,y,z} position of nodes in the map.
4. **Primary channel:** main frequency channel used for transmitting and for carrier sensing.
5. **Transmit power:** default transmit power used for transmitting frames.
6. **OBSS/PD threshold:** sensitivity used within the OBSS/PD-based SR operation.
7. **Received signal strength indicator (RSSI):** average signal quality experienced by STAs during reception phases.
8. **Inter-BSS interference:** average power sensed from devices belonging to other BSSs.

9. **Signal-to-interference-plus-noise ratio (SINR):** average SINR experienced by STAs when receiving data from their AP.

It is worth noting that most of the extracted information is typically obtained on a continuous basis in a real system. Indeed, the RSSI, SINR and throughput measurements can be reported periodically by 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 STAs and APs.

As for the label, we provide the throughput $\gamma_{j,\tau}^{(k)}$ obtained by each STA j in context k during the simulation, provided that the OBSS/PD configuration τ is used. Predicting the throughput is the goal of the implemented FL solutions described in the next section. Notice, as well, that other Key Performance Indicators (KPIs) such as the average delay or the number of SR TXOPs could have been considered.

To conclude this section, we show the correlation matrix between input and output variables in Fig. 4, which is later used as a motivation for some of the proposed ML solutions. 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. Finally, Fig. 5 shows the histogram of some of the most relevant features from the entire dataset.

5. FEDERATED LEARNING SOLUTIONS FOR SPATIAL REUSE

In this section, we describe the solutions proposed by the participants of the 2021 ITU AI for 5G Challenge: *FederationS*, *FedIPC*, and *WirelessAI*.

5.1 FederationS

Our solution is designed in three stages. In the first stage, we analyze and pre-process available datasets. In the second stage, using gained insights from the data analysis, we define a deep neural network (DNN) model running in each client. In the final stage, we describe the proposed FL algorithm.

In the data analysis stage, we consider scenarios *training2* and *training3*, containing 2000 different IEEE 802.11ax deployments (see Table 2 for further details). We extract several features available in the simulator's output files from these scenarios, namely the OBSS/PD configuration, the RSSI, the interference at the reference AP from other APs, the SINR, and the throughput of each STA. We also obtain additional information using available data in the simulator's input files. In particular, we extract the coordinates of APs and STAs to compute the Euclidean distances among them. In addition, we obtain the number of STA served by the reference AP and the number of interfering APs.

To obtain the final dataset to be used to train our model, we pre-process 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 normalization method that transforms all features' in the range $[0, 1]$. Finally, when input data are missing, like when the number of interfering AP 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.

To decide the ML method to be used, we make the following two main observations from the correlation analysis done in Fig. 4:

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, which affects SINR and throughput variables, showing that relationships between input and output values of the system are not straightforward to characterize 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 colors 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 characterized by elements with close to zero correlation values, meaning that the relationship between features is weak. Therefore, these connections are expected to bring a low contribution to the predictions and can be dropped in the

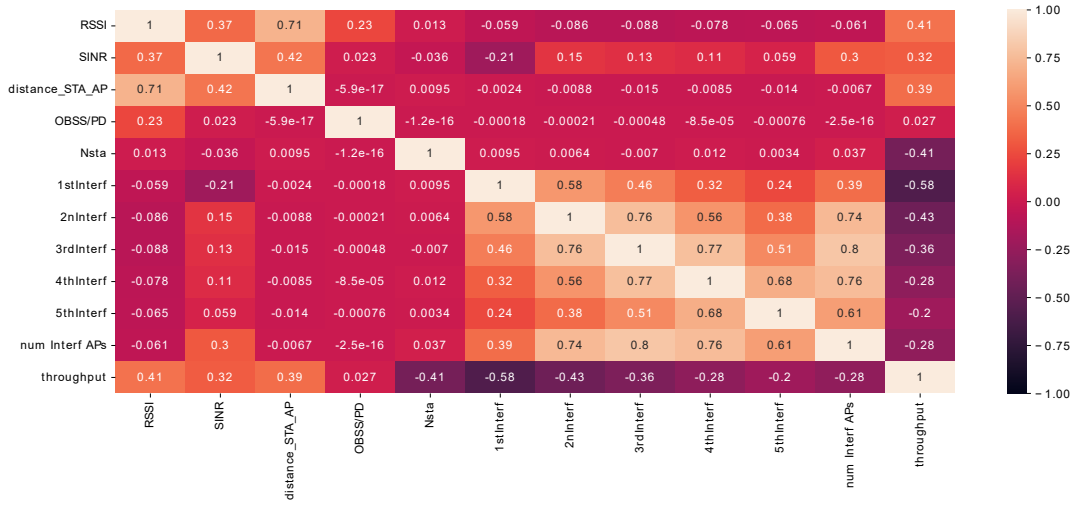


Fig. 4 – Correlation between different input and output variables of the dataset.

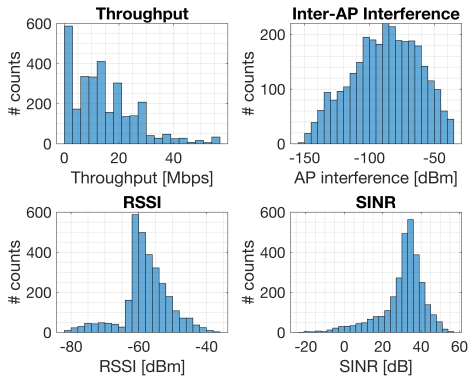


Fig. 5 – Histogram of relevant features from the dataset.

DNN architecture design.

Based on this, in the following design stage, we model DNN architecture as represented in Fig. 6. 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 AP, and the number of interfering AP 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.

As for the FL solution, it is based on the implementation outlined in [52]. In addition, our approach combines the trained weights in a novel way, which is designed

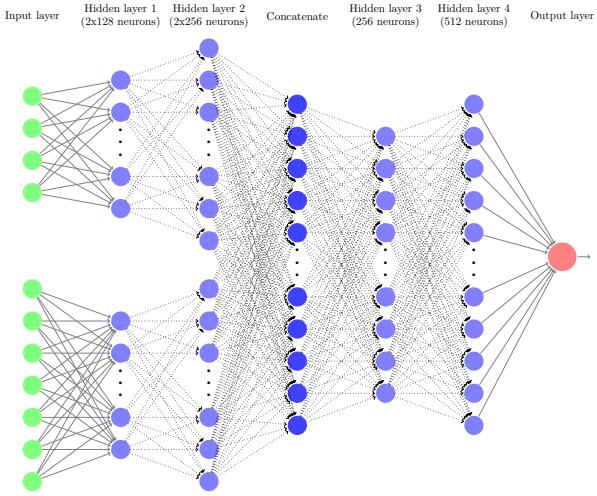


Fig. 6 – Visualization of FederationS' DNN structure.

specifically for this challenge and is based on the data acquired in each context. In particular, our proposed FL solution follows the steps described in Algorithm 2.

In steps 1-5, we initialize the contexts and split them into train and validation sets. After the initialization, 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 behav-

Algorithm 2 FederationS solution.

Init set $\mathbb{K}$, i.e., init K contexts with data samples
From $\mathbb{K}$, select N_{eval} to create $\mathbb{K}_{val}$
Create a new set of contexts $\mathbb{K}_{tr} = \mathbb{K} \cap \mathbb{K}_{val}$
Server initializes model parameters θ_0 and W_0
The server transmits $\theta_0^{(k)}, w_0^{(k)}$ to k-th contexts
for communication epoch $t = 1, 2, ..., T$ **do**
    Rand. select N_{tr} contexts from $\mathbb{K}_{tr}$ to get $\mathbb{K}_{ep}$
    for i-th context in $\mathbb{K}_{ep}$ **do**
        Split samples in β ($\frac{n_i}{B}$ batches of size B)
        Where $n^{(i)}$ is the number of data samples
        for batch b in β **do**
            $\theta_t^{(i)}, w_t^{(i)} \leftarrow \theta_{t-1}^{(i)}, w_{t-1}^{(i)} - \eta \nabla l(\theta_{t-1}^{(i)}, w_{t-1}^{(i)}; b)$
        end for
        Determine weight $\alpha^{(i)} = n^{(i)}/N_{STA}$
        Transmit $\theta_t^{(i)}$, $w_t^{(i)}$, and $\alpha^{(i)}$ to central server
    end for
    Calculate data samples weight $\alpha_t = \sum_{i=1}^{N_{tr}} \alpha^{(i)}$
    Update model: $\theta_t^{(k)} = \sum_{i=1}^{N_{tr}} \frac{\alpha^{(i)} * \theta_t^{(i)}}{\alpha_t}$, $w_t^{(k)} = \sum_{i=1}^{N_{tr}} \frac{\alpha^{(i)} w_t^{(i)}}{\alpha_t}$
    The server transmits $\theta_k^t, w_t^{(k)}$ to k-th contexts
end for
Output: θ_T and w_T

ior 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 normalizes by the number of STA in the contexts. We denote this normalization weights with α. This approach improves the accuracy of the predictions of the throughput.

To evaluate the performance of the proposed solution (shown in Fig. 7), we consider the mean average error (MAE). In particular, we use a neural network trained for $T = 250$ communication rounds. In Table 3, we report the list of hyper-parameters used in the submitted solution and related pre-trained DNN can be found in the GitHub repository [14]. We perform the validation using five percent of available contexts, randomly sampled from $\mathbb{K}$. In total, we had $K = 1946$ available contexts. Five percent of available contexts are used for evaluation, i.e., $N_{eval} = 97$. At each communication round, we select 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 recognize when the solution starts to over-fitting to the training data, i.e., $\mathbb{K}_{val}$, where $\mathbb{K}_{val} \cap \mathbb{K}_{tr} = \emptyset$.

As shown in Fig. 7, 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

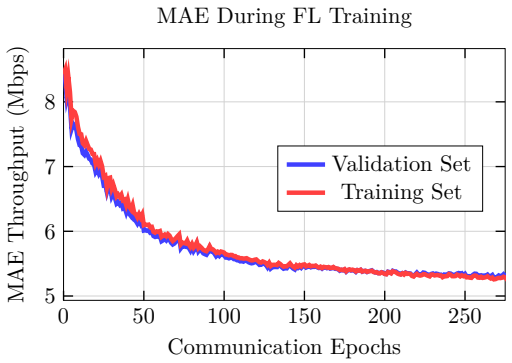


Fig. 7 – MAE obtained by FederationS' over the number of communication epochs.

Table 3 – FederationS hyper-parameters

Neural Network training options	Solver	Adam [53]
	Batch size (B)	21
	Dropout	10%
	Learning rate	10^{-4}
	L2 regularization	10^{-5}

for validation (Validation set). However, the validation throughput decrease is noisier as it sometimes increases between two consecutive communication epochs. Such behavior is due to the random sampling approach as not all randomly selected sets $\mathbb{K}_{ep}$ wholesomely represent the system.

5.2 FedIPC

We design an NN model that predicts the throughput of the STAs of a given BSS for a chosen SR configuration. The goal of this solution is to find the optimal OBSS/PD threshold maximizing the network throughput. To this end, we aim to design a neural network model which predicts the throughput of each STA in the given BSS for a chosen OBSS/PD threshold from a certain range, thus one can tune the OBSS/PD threshold by using NN architecture. In the federated learning setting, we model each context as a node, where their data consist of simulations with different thresholds. We assume these nodes cannot communicate with each other, but communicate with a parameter server in rounds, which aggregates the weights of the nodes to create. Then, the parameter server distributes the updated global model to the clients.

To train the NN model we employ the federated learning framework in the following way; we call a Wi-Fi deployment with specific characteristics such as node locations and number of interfering BSSs as a context. We consider n contexts in total, where for each context, we have s_k STAs per AP for context k. We also define a as the maximum number of access point and b as the maximum number of STAs per AP. Note that all contexts may have different number of STAs per AP. We also have interference sensed by APs, RSSI of the STAs assigned to AP_A,

and the average SINR of each STA in BSS_A. We can control control threshold $\tau_k \in \{-82, -81, \ldots, -62\}$. To simplify the problem as in the competition, we can only change the threshold of the BSS_A, and all other BSSs' thresholds are fixed to -82 dBm. Thus, we only consider the STAs in BSS_A. Let $\gamma_{j,\tau_k}^{(k)} \in \mathbb{R}$ be the throughput of j^{th} STA in the BSS_A of the k^{th} context, where threshold of the BSS_A is chosen as τ_k. For context i, our objective is to find τ_k that maximizes the throughput for all STAs in the BSS_A of context k, i.e,

$$\arg\max_{\boldsymbol{\tau}} \sum_{k=1}^{n} \sum_{j=1}^{s_k} \gamma_{j,\tau_k}^{(k)}, \tag{5}$$

where $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_n]^T$, s_k is the number of STAs connected to each AP (or the number of STAs in each BSS) in context k. Having the knowledge of throughput values $\gamma_{j,\tau'}^{(k)}$, $\forall k, j, \tau'$, which is not likely, one can easily calculate $\boldsymbol{\tau}$ using *(5)*. Thus, to determine the best threshold for each context k, we estimate $\gamma_{j,\tau'}^{(k)}$ via $\hat{\gamma}_{j,\tau_i}^{(k)}$ for all STA j and threshold $\tau^{'}$ combinations.

Since we cannot directly calculate or know the throughput $\gamma_{j,\tau}^{(k)}$, we estimate it via a model $\hat{\gamma}_{j,\tau}^{(k)} = f_{j,\tau}^{(k)}$, where i is the context index, j is the index of STA connected to AP_A and τ is the threshold. Moreover, estimating one STA's throughput is highly related to estimating another STA's throughput in the same context. Thus, to exploit this relation, we formulate the throughput regression problem as multi-output regression, as the following:

$$\boldsymbol{f}_{\tau}^{(k)}(\boldsymbol{x}_{\tau}^{(k)}, \boldsymbol{W}_i) = \left[f_{1,\tau}^{(k)} \; f_{2,\tau}^{(k)} \ldots f_{b,\tau}^{(k)} \right]^T, \tag{6}$$

where k is the context index, $\boldsymbol{x}_{\tau}^{(k)}$ is the input vector and $\boldsymbol{W}_i$ is the neural network weights of the model at context k. The input vector $\boldsymbol{x}_{\tau}^{(k)} \in \mathbb{R}^{4b+a}$ includes each STA's features in order (for STAs in BSS_A). Each STA's features are as the following: interference sensed by APs, RSSI, the average SINR and the threshold, respectively. When a context has less than b STA per AP. We zero pad for the remaining places until the vector reaches the maximum in the dataset. This is possible since a (the maximum number of APs) and b (the maximum number of STAs per AP) are fixed.

Since every context may have different number of STAs per AP, we mask the nonexistent STAs as the following:

$$f_{k,\tau}^{(k)} = \hat{\gamma}_{k,\tau}^{(k)} = \gamma_{k,\tau}^{(k)} = 0, \quad \forall k \in \{s_k + 1, \ldots, b\}.$$

This way, we do not backpropagate any loss for nonexistent STAs and become able to train our model for variable number of STAs per AP for every context. Then, we define the ground truth vector as:

$$\boldsymbol{\gamma}_{\tau}^{(k)} = \left[\gamma_{1,\tau}^{(k)} \; \gamma_{2,\tau}^{(k)} \; \cdots \; \gamma_{b,\tau}^{(k)} \right]^T$$

For the context k (local node), our objective is to minimize mean-squared error for regression task for any $(\boldsymbol{x}_{\tau}^{(k)}, \boldsymbol{\gamma}_{\tau}^{(k)})$ data point among all contexts, i.e.,

$$\arg\min_{\boldsymbol{W}_k} \sum_{\forall \tau, k} \left|\left| \boldsymbol{f}_{\tau}^{(k)}(\boldsymbol{x}_{\tau}^{(k)}, \boldsymbol{W}_i) - \boldsymbol{\gamma}_{\tau}^{(k)} \right|\right|_2^2.$$

We use a feed-forward neural network with one hidden layer as our model $\boldsymbol{f}_{\tau}^{(k)}(\boldsymbol{x}_{\tau}^{(k)}, \boldsymbol{W}_k)$ with weights $\boldsymbol{W}_k = \left[\boldsymbol{W}_k^{(1)} \; \boldsymbol{W}_k^{(2)} \right]$, where $\boldsymbol{f}_{\tau}^{(k)}(\boldsymbol{x}_{\tau}^{(k)}, \boldsymbol{W}_i) = \boldsymbol{W}_k^{(2)} \text{ReLU}(\boldsymbol{W}_k^{(1)} \boldsymbol{x}_{\tau}^{(k)})$, $\boldsymbol{W}_k^{(2)} \in \mathbb{R}^{b \times h}$ and $\boldsymbol{W}_k^{(1)} \in \mathbb{R}^{h \times (a+3b)}$. As seen, we use rectified linear unit (ReLU) as our activation function. Note that this neural network can easily be generalized to a neural network with multiple hidden layers, but in our case, the neural network with only 1 hidden layer have worked the best on the validation set.

To train the proposed NN architecture under the FL paradigm, FedAvg is applied (see Section 3). We consider full participation during FL rounds, meaning that all the users' updates are used for averaging in each communication step. Furthermore, we fix the batch size to $B = 21$ (matching the size of local datasets) and the number of local epochs to $E = 1$. Regarding data splits, we only use the scenario *training3*, as it is the one containing more complex and complete data, and use the 80% of the contexts for training, the 10% of the contexts for the first validation, and the remaining 10% for the second validation. Notice that we use the first validation set for early stopping of the global model, whereas the second one it to perform hyper-parameter tuning. We tune our method by using Tree Parzen Estimator of the Optuna library [54] and choose the model with the lowest MAE on the second validation set.

Finally, we evaluate our global model after every 20 rounds and stop training if no improvement in validation MAE is achieved after $T = 100$ rounds. We evaluate the prediction results of our method via the MAE metric. Recall that we estimate the throughputs by multi-output regression task and each context may have different number of throughputs to be predicted. Thus, we flatten the predictions for existing STAs before calculating the MAE. We normalize the data by minimax normalization. We use the standard SGD implementation of Pytorch [55] to implement federated averaging.

5.3 WirelessAI

We follow the FL framework to address the complex SR problem in multiple 11ax WLAN cells. Individual agents first train their local NN models (with the same network structure) using their local datasets, and then exchange and average model weights through a centralized parameter server.

Typical NN models use simple data structures such as

vectors to encode inputs and outputs. However, in wireless networks characterized by graphs $G = (V, E)$, where V is the set of nodes, and E is the set of wireless links, the number of nodes and the number of links can vary depending on the networking scenarios. It is difficult to fix the vector dimension to fit all networking scenarios. Even if we can fix the dimension and pad zeros to the unused dimension fields, it is meaningless to use these fields.

To overcome the graph representation problem, we treat the whole network as an image. More specially, we first fix a maximum range and treat the whole network as a 1×100×100 gray-scale image with a default value of 0. Then we map nodes to values by their roles (i.e., AP role with value 1, the target AP with OBSS/PDD value, other APs with value 1, and STAs with value 2), and place the values to their corresponding locations. In this way, we can represent any networks with arbitrary APs and STAs. Note that the topology information is encoded into the image.

Then, we adopt two NNs to predict the performance: one part is a convolutional neural network (CNN), and the other part is a fully connected neural network (FCNN). We first use CNN to capture the interactions between STAs and APs to predict the RSSI, SINR of the BSS of interest, and interference to the AP of interest. The input of the CNN is the above processed gray-scale image, and the used OBSS/PD value, and the output of the CNN is the RSSI, SINR, and the caused interference. Then, we use the output of CNN as the input of FCNN to predict the downlink throughput of the AP of interest.

The key rationale of using such architecture is to reduce computation complexity. RSSI, SINR, and the caused interference are the key factors that impact the final performance. Compared with using a whole FCNN to predict the performance, if we can first use CNN to model the relationship among {topology, OBSS/PD value} and {RSSI, SINR, and the caused interference}, and then use a small dimension of FCNN to predict the performance, the computation complexity is reduced.

To empower our FL algorithm, we treat each context as a local client and let each local client use its own data to train the above two NNs. In particular, we follow the standard FL training procedure, and run the training in rounds: the above two NN models are trained by each local client, and the weights of the local models are averaged to generate the global shared model, which is used in the next round. For a dataset, there are overall 1000 local clients, and we randomly choose 10 local clients in each round to generate the average model. The global shared model has been updated using $T = 20$ rounds in total.

The proposed NN model and FL algorithm are implemented in Pytorch.[2] Table 4 and Table 5 summarize the architecture of the proposed NN models. In particular, there are 13 layers in our CNN model including 5 convolution layers, 4 max-pooling layers, 1 adaptive average pooling layer, and 3 fully-connected layers. For each convolution layer, the layers are convolved with kernel size 3. In order to keep the size of the image after each convolution operation and obtain more information on the image edge position, we fill the images (i.e., padding) before each convolution operation. After every convolution layer, a max-pooling operation is applied to the feature maps. The kernel size of the max-pooling layer is 2. The purpose of max-pooling is to reduce the size of the feature maps. The output size of the adaptive average pooling layer is 1. The fully-connected layers consist of respectively 512 and 64 and 17 output neurons. There are 1 input layer, 2 hidden layers, and 1 output layer in our FCNN model. These layers consist of respectively 512 and 128 and 64 and 6 output neurons. The ReLU is used as an activation function for convolution layers and fully-connected layers.

Table 4 – A Summary Table of the Proposed CNN Model.

Layers	Type	Output Size	Kernel Size	Stride
1	Convolution	128×100×100	3	1
2	Max-pooling	128×50×50	2	2
3	Convolution	256×50×50	3	1
4	Max-pooling	256×25×25	2	2
5	Convolution	512×25×25	3	1
6	Max-pooling	512×12×12	2	2
7	Convolution	1024×12×12	3	1
8	Max-pooling	1024×6×6	2	2
9	Convolution	2048×6×6	3	1
10	Adaptive average pooling	1×2048	-	-
11	Fully-Connected	512	-	-
12	Fully-Connected	64	-	-
13	Fully-Connected	17	-	-

Table 5 – Summary Table of the WirelessAI FCNN Model.

Layers	Type	Output Size
1	Fully-Connected	512
2	Fully-Connected	128
3	Fully-Connected	64
4	Fully Connected	6

6. PERFORMANCE EVALUATION

In this section, we show the results obtained by the challenge participants' models presented in Section 5. During the competition, the test dataset was released without revealing the actual throughput of the simulated deployments. Table 6 summarizes the performance accuracy obtained by each participating team on the test dataset.

Next, we analyze the results obtained by each participant with more details. First, Fig. 8 showcases the empirical cumulative distribution function (CDF) of the test error obtained by each solution. The results are

[2]The code used to implement all the proposed methods by WirelessAI is available in Github [16].

Table 6 – Mean average error obtained by the solution proposed by each team.

Team	MAE (Mbps)
FederationS	6.5534
FedIPC	5.8572
WirelessAI	8.913

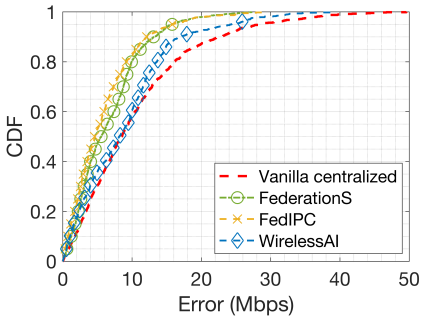


Fig. 8 – Empirical CDF of the error obtained by each participants' solution over the test dataset. A vanilla centralized model is used for comparison purposes.

compared to the ones obtained by a vanilla centralized mechanism, which consists of a feed-forward NN with 1024, 512, and 256 neurons in each of its three layers, with ReLU activation.[3]

As shown, all the proposed FL solutions improve the performance of the naive centralized method. Furthermore, FedIPC is the model providing the highest accuracy, with the 82.4% of the predictions below a 10 Mbps error, compared to the 80.5% and the 60.3% achieved by FederationS and WirelessAI, respectively.

Finally, to provide more insights on the generalization capabilities of each model, FIg. 9 shows the error obtained by each solution, for each possible number of APs and STAs in the test deployments. As shown, contrary to intuition, all the models perform better as complexity increases, i.e., as the number of APs and STAs is bigger. This result is mainly motivated by the fact that DL allows capturing the most complex interactions in dense deployments, which reinforces the role of AI-enabled solutions for network optimization.

7. DISCUSSION

7.1 Contributions

In this paper, we have presented the main results gathered from problem statement *"ITU-ML5G-PS-004: Federated Learning for Spatial Reuse in a multi-BSS (Basic Service Set) scenario"* in the 2021 ITU AI for 5G Challenge. First, we have overviewed the SR problem in IEEE 802.11ax and formulated a novel optimization use case via FL. To evaluate the potential of this solution, we have provided a dataset containing synthetic data on 11ax SR measurements in random deployments.

[3]For further details on the centralized mechanism, refer to the provided open-access repository [56].

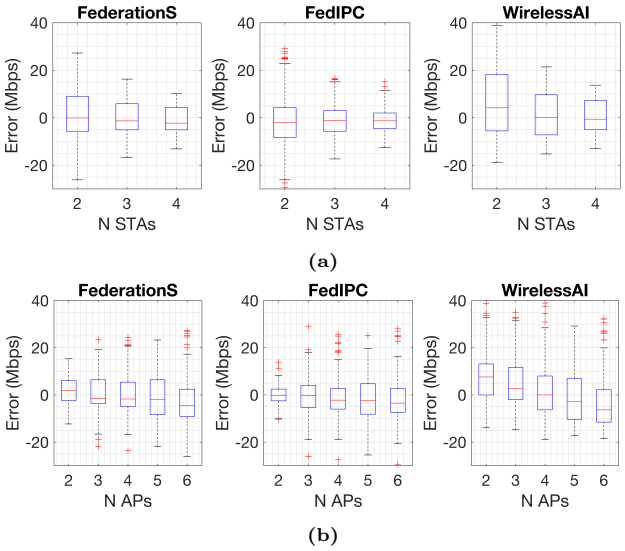


Fig. 9 – Generalization capabilities of the proposed models on test data with respect to: (a) the number of STAs, and (b) the number of APs.

The dataset is open for the sake of reproducibility and to engage other researchers to work on this topic. The provided dataset has been used by the participants of the challenge to develop the models introduced in this paper.

7.2 Lessons learned

We extract the following insights from the models and results overviewed in this paper:

1. Predicting WLAN performance accurately is key to optimize these kinds of networks. However, mechanisms like OFDMA or SR add further complexity to accurately modeling WLANs. In this regard, DL-based models have shown a great potential for capturing the complex interactions of IEEE 802.11ax WLANs applying SR in dense deployments. This provides a paradigm shift with respect to mostly adopted online learning mechanisms. Nevertheless, for the sake of addressing spatial interactions in dynamic WLAN settings, both types of mechanisms are envisioned to be combined.

2. FL suits the decentralized nature of WLANs and, despite contexts count on limited data, its performance has been shown to outperform vanilla centralized ML. Thus, FL provides opportunities for (i) training ML models collaboratively, (ii) enhancing user privacy by not sharing data directly but model weights, (iii) reducing the communication overhead of traditional centralized ML mechanisms, and (iv) providing portability by reducing the computation capabilities for the training of ML models.

3. Finally, using synthetic datasets for training ML models contributes to enriching ground knowledge on certain network technologies and deployments.

Concerning this, we remark the importance of cost-effectively generating data with network simulators, which can complement real networking data to, for instance, reproduce anomalies.

7.3 Future research directions

The SR mechanism is expected to evolve towards a more sophisticated operation in future IEEE 802.11 amendments. At the moment of writing this paper, task group IEEE 802.11be (TGbe) is defining the coordinated SR (c-SR) mechanism as part of the multi-AP operation [57]. Through c-SR, APs collaborate to further improve the performance gains achieved by applying SR. More specifically, APs exchange relevant information (e.g., measurements) to select the best SR configuration, based on the recipient STAs to which transmissions are expected to be held.

Fig. 10 illustrates the basics on c-SR. Considering the deployment shown in Fig. 10a, where two contending APs (namely AP_A and AP_B) are within the same sensitivity area when using the default CCA/CS. Nevertheless, when applying c-SR, both APs can transmit in parallel, thus enhancing spectral efficiency and potentially reducing latency. To do so, the AP gaining channel access after completing the backoff (BO) takes the role of *sharing AP* (in Fig. 10b, AP_A is the sharing AP). Likewise, AP_B acts as a *shared AP*. The sharing AP sends a c-SR Trigger Frame (TF) to indicate the capabilities of the upcoming transmission to selected STA (STA_A), including the maximum acceptable interference level (obtained through measurements). Based on the information provided in the TF, the shared AP decides which STA to transmit and selects the best configuration (e.g., transmit power, MCS) to that end. Notice, as well, that a transmit power limitation is imposed by the sharing AP so that its transmission is not affected by shared APs. Finally, once both simultaneous transmissions take place, a block ACK (BA) is sent by STAs to confirm successful downlink transmissions.

Given the added complexity of evolved SR, the role of ML, and more specifically FL, gains ground. Notice, as well, that in order to set the best configuration that maximizes the overall network performance, both APs and STAs perform measurements related to spectral utilization. Such a rich source of data can be exploited by FL to drive intelligent-based network optimization.

ACKNOWLEDGEMENT

The authors would like to thank enormously everyone that made possible the ITU AI for 5G Challenge, with special mention to Vishnu Ram OV, Reinhard Scholl, and Thomas Basikolo. Likewise, we would like to thank the valuable feedback provided by Dr. Andrea Bonfante. This work has been partially supported by WINDMAL PGC2018-099959-B-I00 (MCIU/AEI/FEDER,UE).

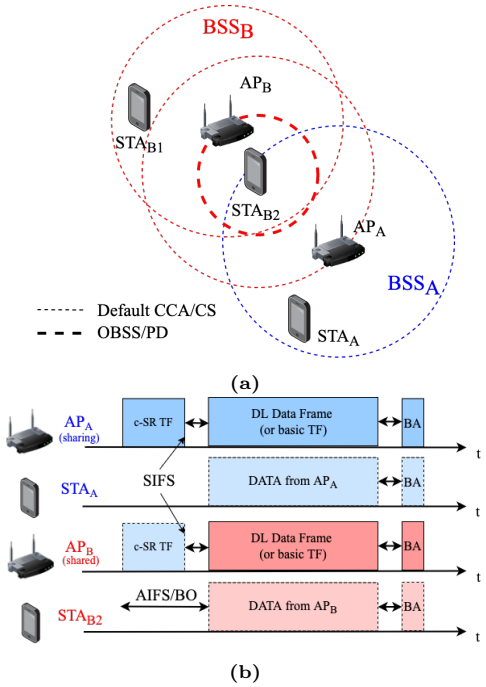


Fig. 10 – c-SR operation: (a) deployment, and (b) exchange of packets.

REFERENCES

[1] Morocho-Cayamcela, M. E., Lee, H., & Lim, W. (2019). "Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions." IEEE Access, 7, 137184-137206.

[2] Akhtar, M. W., Hassan, S. A., Ghaffar, R., Jung, H., Garg, S., & Hossain, M. S. (2020). "The shift to 6G communications: vision and requirements." Human-centric Computing and Information Sciences, 10(1), 1-27.

[3] Klaine, P. V., Imran, M. A., Onireti, O., & Souza, R. D. (2017). "A survey of machine learning techniques applied to self-organizing cellular networks." IEEE Communications Surveys & Tutorials, 19(4), 2392-2431.

[4] Bkassiny, M., Li, Y., & Jayaweera, S. K. (2012). "A survey on machine-learning techniques in cognitive radios." IEEE Communications Surveys & Tutorials, 15(3), 1136-1159.

[5] O'shea, T., & Hoydis, J. (2017). "An introduction to deep learning for the physical layer." IEEE Transactions on Cognitive Communications and Networking, 3(4), 563-575.

[6] Hoydis, J., Aoudia, F. A., Valcarce, A., & Viswanathan, H. (2021). "Toward a 6G AI-Native Air Interface." IEEE Communications Magazine, 59(5), 76-81.

[7] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). "Feder-

ated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492. [Accessed on 20 January 2022]

[8] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., & Cardoso, M. J. (2020). "The future of digital health with federated learning." NPJ digital medicine, 3(1), 1-7.

[9] Du, Z., Wu, C., Yoshinaga, T., Yau, K. L. A., Ji, Y., & Li, J. (2020). "Federated learning for vehicular Internet of things: Recent advances and open issues." IEEE Open Journal of the Computer Society, 1, 45-61.

[10] Brik, B., Ksentini, A., & Bouaziz, M. (2020). "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems." IEEE Access, 8, 53841-53849.

[11] ITU-T. ITU AI/ML in 5G Challenge (2021). Available at: https://challenge.aiforgood.itu.int/. [Accessed on 20 January 2022]

[12] Wilhelmi, F., et al. "Usage of network simulators in machine-learning-assisted 5G/6G networks." IEEE Wireless Communications 28.1 (2021): 160-166.

[13] Wilhelmi, F. (2021). [ITU AI/ML Challenge 2021] Dataset IEEE 802.11ax Spatial Reuse [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.5656866. [Accessed on 20 January 2022]

[14] Hribar, J. & Bonfante, A. (2021). "FederationS: Federated Learning for Spatial Reuse in a multi-BSS (Basic Service Set) scenario". Github repository. Available online: https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-004_Federated_Learning_team_FederarationS. [Accessed on 20 January 2022]

[15] Yilmaz, S. F., Ozfatura, E., Ozfatura, K., Yildiz, O., & Gündüz, D. (2021). "FedIPC Spatial Reuse Project for ITU AI/ML Challenge 2021.". Github repository. Available online: https://github.com/ITU-AI-ML-in-5G-Challenge/fedipc-spatial-reuse. [Accessed on 20 January 2022]

[16] Chen, H., Ye, X., You, L., & Shao, Y. (2021). "WirelessAI: Federated Learning for Spatial Reuse in a Multi-BSS Scenario". Github repository. Available online: https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-004-spatial-reuse-team-WirelessAI. [Accessed on 20 January 2022]

[17] Ziouva, E., & Antonakopoulos, T. "CSMA/CA performance under high traffic conditions: throughput and delay analysis." Computer communications 25, no. 3 (2002): 313-321.

[18] Wilhelmi, F., Barrachina-Muñoz, S., Cano, C., Selinis, I., & Bellalta, B. (2021). "Spatial reuse in IEEE 802.11 ax WLANs". Computer Communications.

[19] Bellalta, Boris. "IEEE 802.11 ax: High-efficiency WLANs." IEEE Wireless Communications 23.1 (2016): 38-46.

[20] López-Pérez, D., et al. "IEEE 802.11 be extremely high throughput: The next generation of Wi-Fi technology beyond 802.11 ax." IEEE Communications Magazine 57.9 (2019): 113-119.

[21] Wilhelmi, F., et al. "Collaborative spatial reuse in wireless networks via selfish multi-armed bandits." Ad Hoc Networks 88 (2019): 129-141.

[22] Wilhelmi, F., et al. (2019). "Potential and pitfalls of multi-armed bandits for decentralized spatial reuse in WLANs". Journal of Network and Computer Applications, 127, 26-42.

[23] Bardou, A., Begin, T., & Busson, A. "Improving the Spatial Reuse in IEEE 802.11 ax WLANs: A Multi-Armed Bandit Approach." Proceedings of the 24th International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. 2021.

[24] Yin, B., et al. "Learning-Based Spatial Reuse for WLANs With Early Identification of Interfering Transmitters." IEEE Transactions on Cognitive Communications and Networking 6.1 (2019): 151-164.

[25] Jamil, I., Cariou, L., & Hélard, JF. "Novel learning-based spatial reuse optimization in dense WLAN deployments." EURASIP Journal on Wireless Communications and Networking 2016.1 (2016): 1-19.

[26] Soto, P., et al. "ATARI: A graph convolutional neural network approach for performance prediction in next-generation WLANs." Sensors 21.13 (2021): 4321.

[27] Wilhelmi, F., et al. "Machine Learning for Performance Prediction of Channel Bonding in Next-Generation IEEE 802.11 WLANs." TU Journal on Future and Evolving Technologies (ITU J-FET), vol. 2, issue no. 5 (2021).

[28] Szott, S., et al. "WiFi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning." arXiv preprint arXiv:2109.04786 (2021). [Accessed on 20 January 2022]

[29] Konečný, J., et al. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016). [Accessed on 20 January 2022]

[30] Nguyen, Dinh C., et al. “Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing.” IEEE Internet of Things Journal (2021).

[31] Long, G., et al. “Federated learning for open banking.” Federated learning. Springer, Cham, 2020. 240-254.

[32] Qu, Y., et al. “A blockchained federated learning framework for cognitive computing in industry 4.0 networks.” IEEE Transactions on Industrial Informatics 17.4 (2020): 2964-2973.

[33] Amiri, MM., & Gündüz, D. ”Federated learning over wireless fading channels.” IEEE Transactions on Wireless Communications 19.5 (2020): 3546-3557.

[34] Lim, WYB., et al. “Federated learning in mobile edge networks: A comprehensive survey.” IEEE Communications Surveys & Tutorials 22.3 (2020): 2031-2063.

[35] Yang, Z., et al. “Federated learning for 6G: Applications, challenges, and opportunities.” arXiv preprint arXiv:2101.01338 (2021). [Accessed on 20 January 2022]

[36] Li, Y., et al. “Privacy-Preserved Federated Learning for Autonomous Driving.” IEEE Transactions on Intelligent Transportation Systems (2021).

[37] Brik, B., Ksentini, A., & Bouaziz, M. “Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems.” IEEE Access 8 (2020): 53841-53849.

[38] Wang, X., et al. “In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning.” IEEE Network 33.5 (2019): 156-165.

[39] Mashhadi, MB., et al. ”Federated mmWave beam selection utilizing LIDAR data.” Wireless Communication Letters (2021).

[40] Khan, LU., et al. “Federated learning for internet of things: Recent advances, taxonomy, and open challenges.” IEEE Communications Surveys & Tutorials (2021).

[41] McMahan, B., et al. “Communication-efficient learning of deep networks from decentralized data.” Artificial intelligence and statistics. PMLR, 2017.

[42] Reddi, S., et al. “Adaptive federated optimization.” arXiv preprint arXiv:2003.00295 (2020). [Accessed on 20 January 2022]

[43] Ozfatura, E., Ozfatura, K., & Gündüz, D. “FedADC: Accelerated Federated Learning with Drift Control.” arXiv preprint arXiv:2012.09102 (2020).

[44] Zhang, C., et al. “A survey on federated learning.” Knowledge-Based Systems 216 (2021): 106775.

[45] Li, T., et al. “Federated learning: Challenges, methods, and future directions.” IEEE Signal Processing Magazine 37.3 (2020): 50-60.

[46] Chen, M., et al. ”Distributed learning in wireless networks: Recent progress and future challenges.” Journal on Selected Areas in Communications (2021).

[47] Turkcell (2022). “Radio Link Failure Prediction dataset”. Github repository. Available online: https://github.com/Turkcell/ITU-AIMLin5GChallenge-2021. [Accessed on 20 January 2022]

[48] Barrachina-Muñoz, S., Bellalta, B., & Knightly, E. W. “Wi-fi channel bonding: An all-channel system and experimental study from urban hotspots to a sold-out stadium.” IEEE/ACM Transactions on Networking 29.5 (2021): 2101-2114.

[49] Suárez-Varela, J., et al. “The graph neural networking challenge: a worldwide competition for education in AI/ML for networks.” ACM SIGCOMM Computer Communication Review 51.3 (2021): 9-16.

[50] Wilhelmi, F. (2020). [ITU-T AI Challenge] Input/Output of project “Improving the capacity of IEEE 802.11 WLANs through Machine Learning” [Dataset]. Zenodo. http://doi.org/10.5281/zenodo.4106127

[51] Barrachina-Muñoz, S., Wilhelmi, F., Selinis, I., & Bellalta, B. (2019, April). “Komondor: a wireless network simulator for next-generation high-density WLANs”. In 2019 Wireless Days (WD) (pp. 1-8). IEEE.

[52] The TensorFlow Federated Authors (2018). “TensorFlow Federated”. Github repository. Available online: https://github.com/tensorflow/federated. [Accessed on 20 January 2022]

[53] Kingma, DP., & Jimmy Ba. “Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014). [Accessed on 20 January 2022]

[54] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). “Optuna: A next-generation hyperparameter optimization framework.” In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2623-2631).

[55] Paszke, A. et al. (2017). Automatic differentiation in pytorch.

[56] Wilhelmi, F., et al. (2021). "ITU_AI_Challenge_2021". Github repository. Available online: https://github.com/fwilhelmi/ITU_AI_Challenge_2021. [Accessed on 20 January 2022]

[57] Nunez, D., Wilhelmi, F., Avallone, S., Smith, M., & Bellalta, B. (2021). "TXOP sharing with Coordinated Spatial Reuse in Multi-AP Cooperative IEEE 802.11 be WLANs." arXiv preprint arXiv:2112.00515. [Accessed on 20 January 2022]

AUTHORS

Francesc Wilhelmi holds a Ph.D. in information and communication technologies (2020) from Universitat Pompeu Fabra (UPF). He is currently a postdoctoral researcher at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) and a teaching assistant at Universitat Oberta de Catalunya (UOC).

Jernej Hribar is a Postdoctoral Researcher at the CONNECT Telecommunications Research Centre, headquartered at Trinity College Dublin. He obtained his PhD from Trinity College Dublin in 2020. He is investigating the use of Artificial Intelligence (reinforcement learning, intelligent agents, and multi-agent systems) in large-scale IoT deployments. His research interests include Age of Information, Federated Learning, Deep Reinforcement Learning, and Embedded Systems.

Selim F. Yilmaz received his B.S. in computer engineering and M.S. degree in electrical and electronic engineering from Bilkent University (Turkey), in 2019 and 2021, respectively. He is currently pursuing his Ph.D. degree at Imperial College London, UK, where he is a member of the Information Processing and Communications (IPC) Lab. His current research interests include edge learning and inference, federated learning, and anomaly detection.

Emre Ozfatura received his B.Sc. in Electronics Engineering with Math minor and M.Sc. in Electronics Engineering from Sabanci University (Turkey), in 2012 and 2015, respectively. He is currently pursuing his Ph.D. degree at Imperial College London, UK, where he is a member of the Information Processing and Communications (IPC) Lab. His research interests are video streaming applications, distributed content storage and distributed computation.

Kerem Ozfatura received BSc from from Istanbul Ozyegin University and currently he is a visiting researcher at Imperial College London, IPC lab. His research areas are federated learning, adversarial training and robust learning.

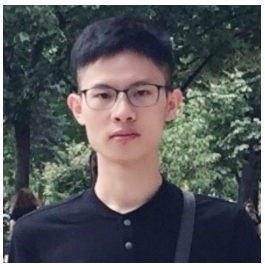
Ozlem Yildiz received her B.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey, in 2020. She is currently a Ph.D. student in the Department of Electrical and Computer Engineering at NYU Tandon School of Engineering. Her research interests are in wireless communications, information theory, mmWave frequencies.

Deniz Gündüz received the B.S. degree in electrical and electronics engineering from METU, Ankara, Turkey, in 2002, and the M.S. and Ph.D. degrees in electrical engineering from NYU Polytechnic School of Engineering (formerly Polytechnic University), Brooklyn, NY, in 2004 and 2007, respectively. He is currently a Professor of Information Processing in the Electrical and Electronic Engineering Department of Imperial College London, and is leading the Information Processing and Communications Lab.

Hao Chen received the B.S. degree in Internet of Things Engineering, in 2021, from the School of Physics and Information Engineering of Fuzhou University, Fuzhou, China. He is currently working towards the M.S. degree in Information Engineering at the School of Informatics, Xiamen University, Xiamen, China. His research interests include wireless communications.

Xiaoying Ye received the B.S. degree in Communication Engineering, in 2021, from the School of Physics and Information Engineering of Fuzhou University, Fuzhou, China. She is currently working towards the M.S. degree in Information Engineering at the School of Informatics, Xiamen University, Xiamen, China.

Lizhao You received his B.S. and M.E. degrees from Nanjing University, China, in 2009 and 2013, and the Ph.D. degree from The Chinese University of Hong Kong, China, in 2016. He joined Huawei Technologies Co., Ltd. after his graduation until 2021. He is currently an Assistant Professor at Xiamen University, China. His research interests include wireless communications, computer networks and network verification.



Yulin Shao received his B.S. and M.S. degrees from Xidian University (Hons.) in 2013 and 2016, and the Ph.D. degree from The Chinese University of Hong Kong (CUHK) in 2020. He was a research assistant with the Institute of Network Coding, a visiting scholar at Massachusetts Institute of Technology, and a postdoctoral fellow at CUHK. He is currently a research associate at Imperial College London. His research interests include wireless communications and machine learning.

Paolo Dini received M.Sc. and Ph.D. from the Universit'a di Roma La Sapienza, in 2001 and 2005, respectively. He is currently a Senior Researcher with the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). His current research interests include sustainable networking and computing, distributed optimization and optimal control, machine learning, multi-agent systems and data analytics.

Prof. Dr. Boris Bellalta is the head of the Wireless Networking research group at the Department of Information and Communication Technologies at Universitat Pompeu Fabra. His research interests are in the area of wireless networks, adaptive systems and machine learning, with emphasis on the design, modelling and performance evaluation of spectrum access protocols.