Background

Machine learning techniques such as Deep Learning (DL) are envisioned to address the complex spatial reuse (SR) problem in multiple 11ax WLAN cells. To keep data confidentiality, federated learning (FL)-based training and inference is preferable. In this competition, we are to use the FL technique to train an DL model, and use that model to predict the downlink throughput of a particular Access Point (AP) with a specific OBSS/PD parameter value in typical dense environments.

Our Designs

NN Model: Typical neural network (NN) models use simple data structure 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 default value 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 Convolutional Neural Network (CNN), and the other part is Fully Connected Neural Network (FCNN). We first use CNN to capture the interactions between STAs and APs to predict the RSSI, Signal to Interference plus Noise Ratio (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.

Federated Learning 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 generated the average model. The global shared model has been updated 20 rounds in total.

Implementation: The proposed NN model and FL algorithm are implemented in Pytorch. Table 1 and Table 2 summarize the architecture of the proposed NN model. 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 of 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 max-pooling layer is 2. The purpose of max-pooling is to reduce the size of the feature maps. The output size of 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 rectifier linear unit (ReLU) is used as an activation function for convolution layers and fully-connected layers. Our implementation is available in Github [1].

Reference

[1] https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-004-spatial-reuse-team-WirelessAI

Table1 A summary table of the proposed CNN model

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

Table2 A summary table of the proposed FCNN model

Layers	Type	Output size
1	Fully-connected	512
2	Fully-connected	128
3	Fully-connected	64
4	Fully-connected	6