Multi-Layer Perceptron for OBSS performance prediction: Predicting the throughput of IEEE 802.11 WLANs

Ramon Vallés Puig
UPF/ITU-T AI Challenge (ML5G-PS-013)

Abstract — One of the problems that arise when planning a configuration of a certain wireless environment, is the difficulty to predict the efficiency of such a scenario before being implemented. This issue takes on major importance when there is some interest in maintaining a certain level of QoS, and a configuration made without the guarantee of its efficiency may not be optimal for what the actual service demands. In this research we seek to implement a ML approach to be able to predict the performance of a network composition where we do not know how the configuration will react.

1. Introduction

The most common way, when it comes to evaluate the performance of a network in a given deployment, is by measuring its throughput. The throughput can vary depending on many factors. The physical layer in a network communication is one of those. Wireless local area networks, when placed near other wireless devices, may suffer from interferences that can disturb the communication between the Access Point and the Stations connected to it, which results in a low throughput and therefore a poor connection performance.

Another interference factor is the so-called Overlapping Basic Service Sets (OBSS), which happens when two or more BSS try to transmit on the same channel (or adjacent channel) at the same time without respecting the transmission shifts. This usually occurs in a practice commonly used by IEEE 802.11 WLANs, called Channel Bonding [1], that consist of combining the maximum number of adjacent channels in a BSS to increase transmission capacity. Thus, in these cases the initial objective of taking advantage of the maximum bandwidth is left behind, and both end up interfering with each other.

In this research we are going to exploit the power of Multi-Layer Perceptron algorithms so that we can predict the aggregate throughput of a BSS in a crowd scenario, where different Access Point may suffer from OBSS. Moreover, we will make a brief analysis of the results and propose alternatives to improve the actual solution.

2. Dataset

All the data used during this research has been provided by the UPF and generated with the Komondor [2] wireless network simulator.

The provided dataset was composed by 2 main different sets of scenarios, with 8 and 12 APs respectively, and each formed by 3 subsets with a slightly variation on the map size.

Scenario 1 (12 APs, 10-20 STAs):

- 1a (80 x 60 m): 100 random deployments
- 1b (70 x 50 m): 100 random deployments
- 1c (60 x 40 m): 100 random deployments

Scenario 2 (8 APs, 5-10 STAs):

- 2a (60 x 40 m): 100 random deployments
- 2b (50 x 30 m): 100 random deployments
- 2c (40 x 20 m): 100 random deployments

2.1 Pre-Processing Data

The dataset contained a huge amount of information, many of which was not relevant for the problem this research aims to solve. Some of the data that has been collected are, for each device: the type of device it is, the coordinates in which it is located, the minimum and maximum channel through which it is allowed to transmit. And regarding the data obtained by the simulation, the Received Signal Strength Indicator (RSSI), and the Signal-to-Interference-plus-Noise Ratio (SINR) per station, the amount of time it has been transmitting by each channel of the AP (Airtime), and the resulting throughput has been taken.

Once the information from each device was selected, the first challenge encountered was the pre-

processing data phase. This is a very delicate phase since the way we prepare the features will have a big impact when training our model.

2.2 Data cleaning

It is often the case, when working with large amounts of information, that among the values we have in our dataset some are corrupt, or simply have no value at all. In our case, these values can be found, for instance, when collecting the received power by each device, where each Access Point will receive an infinite amount of power on itself. Another case could be found when a STA receives so much interference that the SINR is completely obfuscated and it results in a null value or NaN (Not a Number). In this case, the NaN values must be inputted as negative SINR (between [0, -20]).

2.3 Data Transformation

One could think, that the best way to face this problem is treating the stations individually, analyzing their behavior together with the others and predicting which would be the throughput of each one to, later on, calculate the aggregate throughput (given by the sum of the throughput of all the stations that transmit through the same AP). This would be an ideal way to solve the problem. But things get complicated when the scenario uses a dynamic channel bonding policy [3] (as we do). In this case, at each moment of time, every station agrees with the AP to transmit on the maximum number of channels available, which may vary depending on the interference received by the AP on the different channels. At certain times, your station may not be able to transmit on any of the available channels. This phenomenon is called **contention** and happens when all the channels from the same AP are receiving interference from neighbors, if this happens, the station will have to wait until one or more channels are free again to transmit.

As we can see, there is no clear way to predict the bandwidth that will be used by one station with respect to another, as it depends on external factors plus the order in which they start transmitting, determined by the CSMA protocol [4].

Since this model intends to treat all the stations as one, this problem of contention will be reduced, because we will not consider which station is transmitting at any given time. Instead we will consider only the simple fact that the AP transmits all the time (or the time indicated by the Airtime feature), regardless of the connectivity order of the stations.

The idea then, is to compute the distribution of the STAs radio quality, and space distances to its AP, so that we englobe the situation as if there were a single STA.

E.g.: Let's suppose two APs (AP1 and AP2) with 2 and 4 devices respectively whose SINR values are AP1 [27,31], AP2 [3,4,15,26]. As we are interested in having a fixed number of features to train our model, and the number of devices may vary, we are going to encode the properties of the different devices according to the properties of their distribution (mean, standard deviation, num of STAs). In this way, the properties of the stations will be represented by a fixed number of features. (e.g. AP1:[mean:29, std:1.41, n:2] AP2:[mean:12, std:4.68, n:4])

This method has been applied to 3 main components of the Radio Signal which are the previously mentioned SINR, the RSSI, and the distance from each device to the corresponding AP.

We are also interested in representing the amount of channels available, to do so, we will create a size 8 vector, where each position will correspond to a channel, and the value it contains will represent the amount of time it has been transmitting on that channel (Airtime).

Regarding the throughput: as we do not consider the stations individually, we only keep the total aggregate throughput of each AP, and discard the throughput per STA. The resulting dataset will look like this: (for each AP in each scenario):

- **Features:**
  - [0-1] SINR [mean, std]
  - [2-3] RSSI [mean, std]
  - [4-5] Distance [mean, std]
  - [6] Number of stations
  - [7-14] Airtime per Channel
- **Target:**
  - [0] AP Throughput

3. Proposed Model

Once the dataset is ready to be used, the second phase in which we find ourselves is “how” the selected information should be used to obtain the maximum accuracy for new predictions. For this, it has been considered appropriate to design a model based on the already known Multi-Layer Perceptron.

To address the problem, we must first identify which data in the dataset are correlated (Fig.1) and select them carefully to avoid confusing our system.

With this in mind, we are going to divide the neural network in 3 parts: the first one will be in charge of measuring the signal quality of each AP with respect to its stations, the second one is focused on analyzing the available channels of the Access Point through which it can transmit, and the third and last part, takes both outputs from the Signal quality and AP Airtime and computes the overall
result (Fig. 2).

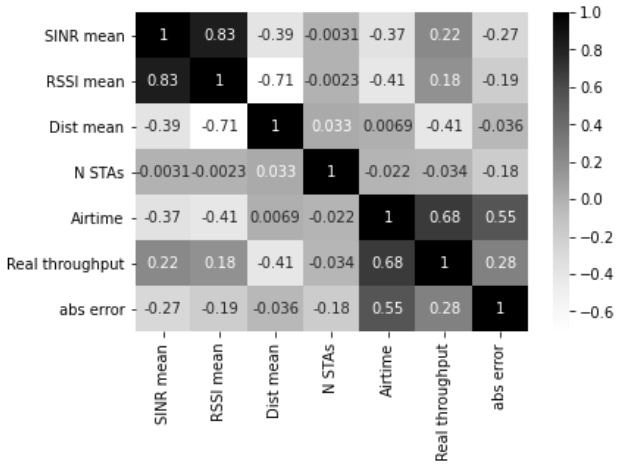

Figure 1: Correlation matrix of the selected features plus the real throughput of the simulation and the absolute error of the proposed model predictions.

3.1 Signal Quality

In this part, the system must understand the distribution of the stations, that is, how dispersed the devices are, what is the power that reaches these devices, and how much interference they receive from their neighbors.

For this reason, the first layer must treat these features (SINR, RSSI, DIST) separately, so that the system understands each possible distribution. For this, we consider 2 separate blocks consisting of a linear

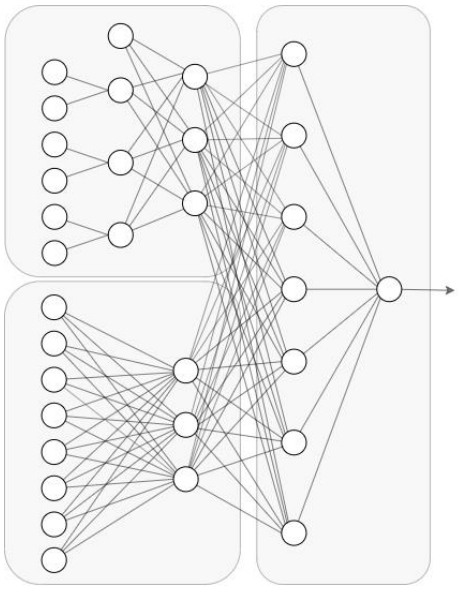

Figure 2: Architecture of the proposed Neural Network Composed by 3 blocks, Radio Signal, Airtime and throughput.

transformation with 2 input features, 1 output, a Parametric Rectified Linear Unit (PReLU) as the activation function and a 1-Dimensional Batch Normalization.

The second layer aims to study the information of the previous distributions, so the model is formed by a second linear layer with 4 neurons; the previously computed outputs and the number of STAs.

3.2 Available channels & Airtime

This part of the neural network is quite straightforward, it consists of a single linear layer, that gets as an input 8 features corresponding to each of the channels Airtime in percentage (%). This layer outputs a 3-dimensional array and is activated with a PReLU function.

3.3 Throughput Prediction:

Once the system has analyzed the radio signal of the AP in respect to its stations, and the airtime in which each channel has been transmitting, we can take this information and pass it through 2 fully connected hidden layers in the neural network that will combine the data to predict the final throughput of the AP. For this final layer, we will be using a ReLU activation function instead. This is because we are not interested in having negative throughputs, so all negative values will be set to zero.

$$PReLU(y) = \begin{cases} y, & if\ y > 0 \\ a \cdot y, & if\ y \leq 0 \end{cases}$$

$$ReLU(y) = \begin{cases} y, & if\ y > 0 \\ 0, & if\ y \leq 0 \end{cases}$$

4 Training

Now that we have implemented the MLP, it is time to train it. To do so, the dataset has been split into two, the first one for training (80%) and the second one for validation (20%).

The training was performed with an evaluation criterion based on the Root Mean Squared Error (RMSE) and the optimizer chosen to optimize the training process is Adam, which has proven to be efficient for this type of ML algorithms.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y} - y_i)^2}{n}}$$

For the training phase, several experiments had been made by modulating the hyperparameters manually. The best results seemed to appear when the learning rate fell close to 0.025 and the number of epochs over 700.

A curious fact that was found during the experiments is that adding the term weight decay, with the idea of regularizing the weights and thus avoiding a possible case of overfitting, resulted in a significant deterioration of accuracy in the vast majority of cases. This might happen because of the simplicity of the model. As it has the appropriate number of neurons and layers, adding a weight decay factor just disturbs the process of learning.

5 Testing

To finally evaluate the results obtained, Pompeu Fabra University has provided us a new dataset with 4 different scenarios each with 50 different deployments. All mapped with a size of 80x60m.

The results of the System Testing are the following:

Scenario 1 (4 APs): TOTAL RMSE: 21.12 Mbps
Scenario 2 (6 APs): TOTAL RMSE: 19.18 Mbps
Scenario 3 (8 APs): TOTAL RMSE: 15.92 Mbps
Scenario 4 (10 APs): TOTAL RMSE: 17.38 Mbps

Each of the scenarios is slightly different, it has been simulated the same way as the training Dataset (using the Komondor simulator) but with varying amount of STAs and APs. The map size is the same for the 4 scenarios, but as the number of Aps increases, so does the density (APs/m^2).

Figures 5, 6, 7 and 8 show, for each scenario, how good the prediction is in relation to reality. These are represented around a linear function, so the further our result is from the line, the worse our prediction has been. This line also divides between those cases in which our system underestimates or overestimates the quality of the signal in each AP (if a value is above the line, it means that our prediction is below the real one, and vice versa).

Epochs	LR	Training loss	Validation loss
700	1E-2	13.14	13.25
500	1E-2	13.10	14.21
700	**2E-2**	**12.69**	**12.96**
500	2E-2	13.77	13.86
700	4E-2	13.15	13.48
500	4E-2	12.84	14.70

Figure 3: Hyperparameters tuning experiments.

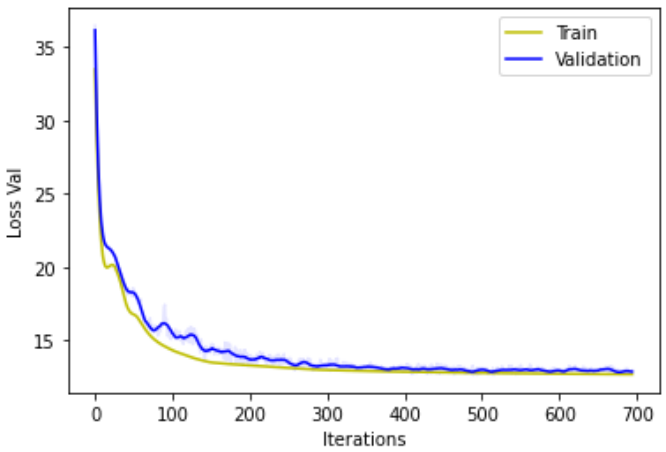

Figure 4: Training Loss curve with LR: 2E-2 and 700 epochs. (80% Training, 20% Validation).

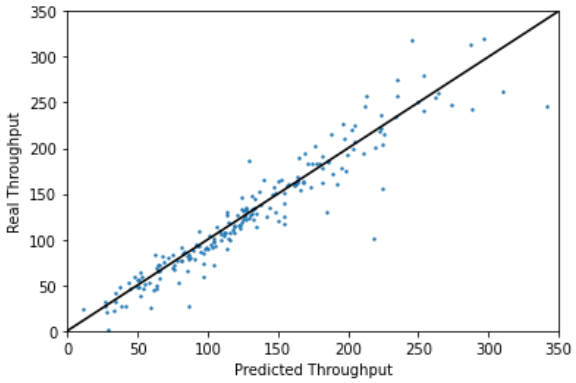

Figure 5: Predictions for scenario 1

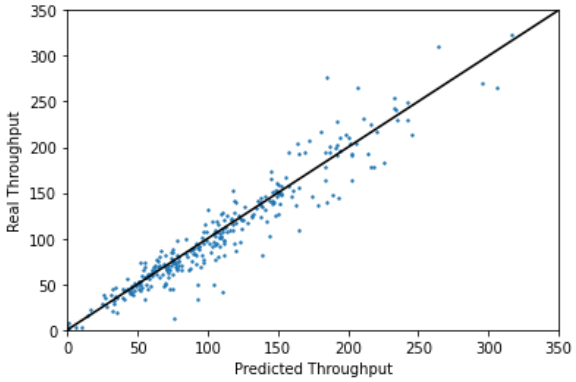

Figure 6: Predictions for scenario 2

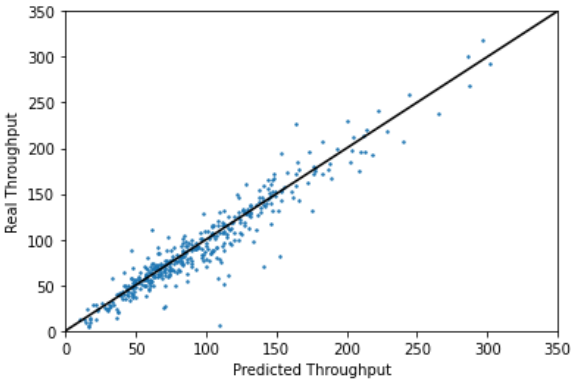

Figure 7: Predictions for scenario 3

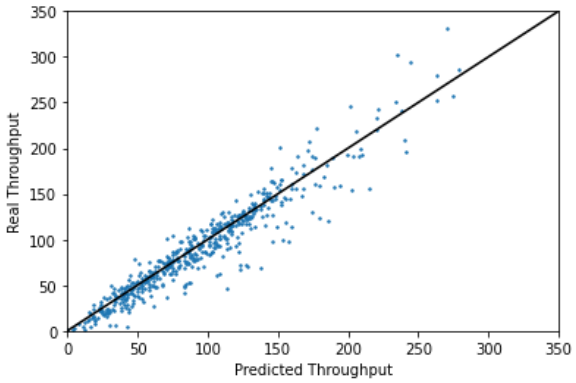

Figure 8: Predictions for scenario 4

6. Discussion and future work

We can therefore confirm that the designed neural network offers quite accurate predictions. However, it seems that, for some of those scenarios that our model has not previously seen (scenario 1, 2, 4), a decrease in the precision of the predictions is perceived. This may be due, among other things, to a loss of information during the Data Transformation phase previously explained, where it might happen that the distribution of the SATs and their signal are very dispersed, or the number of STAs is very small. In those cases, the average and standard deviation would not be enough to represent the signal of the set of devices.

A possible solution to this problem would be to add more parameters to the distribution, such as the minimum, maximum, some per-centiles or even the peak of the distribution by means of the kurtosis function.

Another solution, more complex but that would probably give more precise results and that could be implemented for future work, would be to leave distributions aside and modify the Neuronal Network so that instead of receiving a fixed number of features, it can deal with a dynamically sized input, and thus deal with the interference between APs. We could see this as a directed graph where nodes (APs) are connected by edges (Interferences). This would give a more precise information to the model about the contention that STAs may suffer. This solution could be easily implemented with the concept of Graph Neural Network (GNN)

7. Conclusion:

We have described a deep learning algorithm (MLP) capable of approximating the performance of an OBSS [5]. We have analyzed the results with scenarios of different density and pointed out some limitations such as the loss of relevant information from the SINR and RSSI signal in the Data Transformation phase. Finally, we have proposed possible innovative improvements for the algorithm in question that could improve the accuracy of our predictions in a future project.

References

[1] Barrachina-Muñoz, S., Wilhelmi, F., & Bellalta, B. (2019). *Dynamic channel bonding in spatially distributed high-density WLANs. IEEE Transactions on Mobile Computing.*

[2] 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). IEEE*. https://github.com/wn-upf/Komondor

[3] Barrachina-Muñoz, S., Wilhelmi, F., & Bellalta, B. (2019). To overlap or not to overlap: Enabling channel bonding in high-density WLANs. Computer Networks, 152.

[4] N. Shenoy, J. Hamilton, A. Kwasinski and K. Xiong, "*An improved IEEE 802.11 CSMA/CA medium access mechanism through the introduction of random short delays*," 2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Mumbai, 2015, pp. 331-338.

[5] Source code of the proposed MLP model (open-source) https://github.com/VPRamon/MLP-Throughput-prediction

[6] International Telecommunication Union (ITU) official webpage: https://www.itu.int/

[7] Google Colab service used to train and test the model on the cloud: https://colab.research.google.com/

Annotations

This project is part of a challenge promoted by Universitat Pompeu Fabra (UPF) and is part of the ITU Artificial Intelligence/Machine Learning in 5G Challenge [6].

The project has been implemented in python using the pytorh library and it has been tested in the Google Collaboratory Servers [7] to speed up the executions thanks to the free available GPUs in the cloud.