A Python Tool to Predict Wireless Network Signals in Indoor Environments using Neural Networks

Breno Batista Nascimento, Silva, and Edson Tafeli C.Santos

***Abstract* — The use of neural networks proved to be effective in creating more accurate predictive models compared to traditional approaches. The Python tool developed made it possible to train and adjust these models based on the information collected, taking into account factors such as the physical structure of the site, obstacles present and building materials. The results obtained during the research indicated significant improvements in prediction accuracy compared to conventional methods. This suggests great potential for the practical use of the tool in real-world scenarios, such as the planning and optimisation of indoor wireless networks, contributing to more stable and reliable connectivity indoors. The aim of this work was to create a Python-based tool that uses neural networks to predict wireless network signals in indoor environments. The innovative approach, which combines mapping and field measurements, demonstrated an increase in the accuracy of predictions, promoting advances in the efficiency and reliability of wireless networks in indoor spaces.**

***Index Terms* — Signal prediction , propagation model , neural networks , perceptron's .**

I. INTRODUCTION

The widespread use and availability of wireless networks has revolutionized modern connectivity, enabling seamless data transmission and interaction between devices. However, the complex challenges associated with signal propagation in indoor environments, characterised by obstacles, interference, and signal attenuation, represent obstacles to ensuring consistent and reliable performance of wireless networks. This work addresses these challenges by introducing an innovative tool created using the Python programming language. The tool uses neural networks to improve the prediction and accuracy of wireless network signals, based on measured data for mapping and training on real-world measurements.

This contribution focus on a comparative analyses of path loss propagation models in indoor industrial environments at 2.4 GHz Industrial, Scientific and Medical (ISM) band and 5.0 GHz, Wi-Fi.

This work was supported in part by the Mack-Pesquisa Program at U.PM, which financed equipment and scholarships for the authors.

Breno Batista Nascimento Silva is an undergraduate student in Electrical Engineering at the U.P.Mackenzie School of Engineering (E.E) -São Paulo/SP-Brazil. (batistabreno03@gmail.com)

The main objective of this article was to develop a tool based on an empirical propagation model to provide first-order coverage prediction results in indoor environments using low-cost tools.
The starting point of this work is the one-slope model for training the neural network that was implemented.

II. PATH LOSS PROPAGATION MODELS

Radio frequency signals are the main mechanism for propagating information. The basic model of radio propagation is based on the transmitter and receiver and the transmission medium. Propagation in confined media occurs when the electromagnetic wave passes through a material medium in a closed environment, thus limiting a region of space where multiple reflections of signal infractions can occur.

A. One – Slope Model

The path loos in dB is given by

$$L_{dB}= L_{0,dB} +10n\log(d) \qquad (1)$$

where L0,dB is the path loss obtained at distance of 1.0 m from the transmitter and path loss exponent n is determined

TABLE I – L(D0) FOR VARIOUS VALUES AND FREQUENCIES

L(d0) (dB)	
Frequency (MHz)	L0
900	31.5
1900	38.0
2400	40.2
4000	44.5
5300	46.9

E.T.C.Santos and professor of Electrical Engineering at U.P.Mackenzie in the Digital TV laboratory of the School of Engineering (E.E) -São Paulo/SP-Brazil. (edson.santos@mackenzie.br).

experimentally using a linear interpolation procedure [1].

III. NEURAL NETWORKS

Artificial Neural Networks (ANNs) are data structures based on the functioning of the human brain, it is a bio-inspired computational model, this data structure is made up of artificial neurons, which are inspired by natural neurons. The brain is a highly complex, non-linear and parallel computer (information processing system). It has the ability to organize its structural constituents, known as neurons, in such a way as to carry out certain processing (e.g. pattern recognition, perception and motor control) much faster than the fastest existing computer[2].

Neural networks have a network of artificial neurons that are interconnected, and through Learning Algorithms, simulate the decision-making capacity of the human brain. A neural network is a massively parallelized processor made up of simple processing units that have the natural propensity to store experiential knowledge and make it available for use.
It resembles the brain in two respects:

a) Knowledge is acquired by the network from its environment through a learning process.
b) Connection strengths between neurons, known as synaptic weights, are used to store the acquired knowledge.

The learning process of ANNs is one of the important qualities of these structures. The term "learning" corresponds to the process of adjusting the network's free parameters through a mechanism of presenting environmental stimuli, known as input or training patterns (or data): Stimulus -> adaptation -> new network behaviour.
There are basically three learning paradigms:

Supervised learning: also known as teacher learning, in which the teacher has knowledge of the environment and provides the desired input-response example set. Training is done using the error correction learning rule.

i. Unsupervised learning: there is no supervisor to evaluate the network's performance in relation to the input data. No error measure is used to feed back to the network. They generally employ a competitive learning algorithm (the network's output neurons compete to become active, with a single neuron winning the competition).

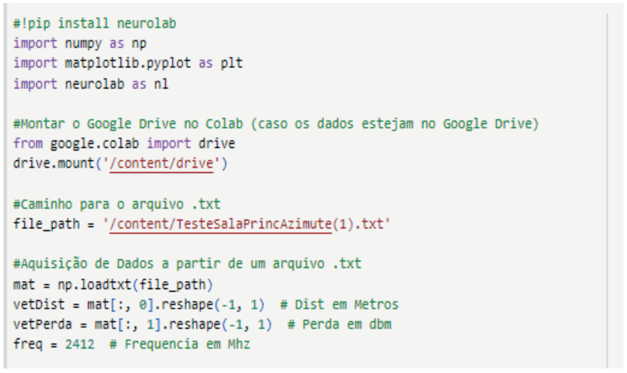
ii. Reinforcement learning: there is no direct interaction with a supervisor or specific model of the environment. Generally, the only information available is a scalar value that indicates the quality of the ANN's performance. During the learning process, the network tests some actions (outputs) and receives a reinforcement signal (stimulus) from the environment that allows it to evaluate the quality of its action.

IV. METHODOLOGY

The procedures of this study were structured in different stages, with the aim of evaluating the effectiveness of the Multilayer Perceptron Artificial Neural Network (MLP ANN) in predicting Wi-Fi signal loss in indoor environments. The stages were outlined as follows:

A. Creation and Training of the RNA-MLP

At this stage, the RNA-MLP was created and configured using the neurolab library in Python. The data collected, containing information on distance and Wi-Fi signal loss, was used to train the neural network. The structure of the RNA-MLP consisted of input layers, one or more intermediate layers and an output layer. Training was carried out using the gradient descent algorithm to adjust the network's synaptic weights, minimizing the prediction error.



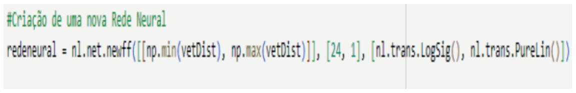
```
#!pip install neurolab
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl

#Montar o Google Drive no Colab (caso os dados estejam no Google Drive)
from google.colab import drive
drive.mount('/content/drive')

#Caminho para o arquivo .txt
file_path = '/content/TesteSalaPrincAzimute(1).txt'

#Aquisição de Dados a partir de um arquivo .txt
mat = np.loadtxt(file_path)
vetDist = mat[:, 0].reshape(-1, 1)  # Dist em Metros
vetPerda = mat[:, 1].reshape(-1, 1)  # Perda em dbm
freq = 2412  # Frequencia em Mhz
```

Fig. 1 Data import via txt file in python Collab.

The code extract in question describes the creation, configuration, training and evaluation of an Artificial Neural Network (ANN) using the "neurolab" library in Python. The neural network is designed to predict Wi-Fi signal losses based on the distances between measurement points and the corresponding signal losses.

B. Creation of the Neural Network

The neural network is initialised using the newff() function from the "neurolab" library. In this case, the network is configured with an input layer, an intermediate layer with 24 neurons and an output layer with 1 neuron. The minimum and maximum distance ranges (vetDist) are supplied as input to the network's input layer.

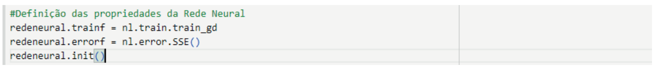


```
#Criação de uma nova Rede Neural
redeneural = nl.net.newff([[np.min(vetDist), np.max(vetDist)]], [24, 1], [nl.trans.LogSig(), nl.trans.PureLin()])
```

Fig. 2 Create Neural network

C. *Definition of Neural Network Properties*

The neural network is configured to use the gradient descent training algorithm (train_gd) and the sum of squares error function (SSE()) to evaluate the prediction error. In addition, the neural network is initialised.

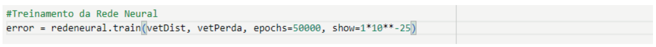


```
#Definição das propriedades da Rede Neural
redeneural.trainf = nl.train.train_gd
redeneural.errorf = nl.error.SSE()
redeneural.init()
```

Fig. 3 Definition of the Neural Network's properties.

D. *Neural Network Training*

The training stage is carried out using the neural network's train() method. In this case, the distance data (vetDist) and the corresponding signal losses (vetLoss) are used to train the neural network. Training is conducted for a specific number of epochs (in this case, 50,000 epochs) and the error value is displayed periodically (every 1x10e-25).

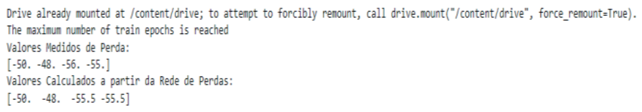


```
#Treinamento da Rede Neural
error = redeneural.train(vetDist, vetPerda, epochs=50000, show=1*10**-25)
```

Fig. 4 Neural Network Training

E. *Comparison to Field Tests*

At this stage, the results obtained by the RNA-MLP were compared with real data collected through field tests. The field tests involved directly measuring Wi-Fi signal strength at different distances within the environment. The comparison sought to verify the ability of the MLP-NRNA to accurately predict signal loss compared to practical observations.

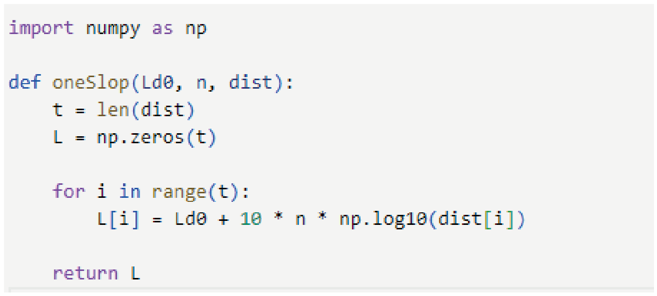


```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
The maximum number of train epochs is reached
Valores Medidos de Perda:
[-50. -48. -56. -55.]
Valores Calculados a partir da Rede de Perdas:
[-50.  -48.  -55.5 -55.5]
```

Fig. 5 Values collected in field measurements.

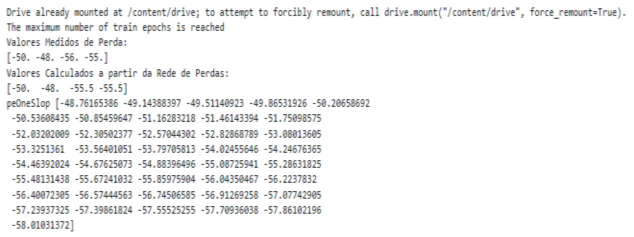
F. *Comparison with the OneSlope Model*

This stage involved comparing the results obtained by the RNA-MLP with the values calculated using the OneSlope propagation model. The OneSlope model is an analytical approach to estimating signal loss in indoor environments. The aim of the comparison was to assess the ability of the RNA-MLP to overcome the limitations of the traditional analytical model and provide more accurate predictions.



```
import numpy as np

def oneSlop(Ld0, n, dist):
    t = len(dist)
    L = np.zeros(t)

    for i in range(t):
        L[i] = Ld0 + 10 * n * np.log10(dist[i])

    return L
```

Fig. 6 One Slope function.

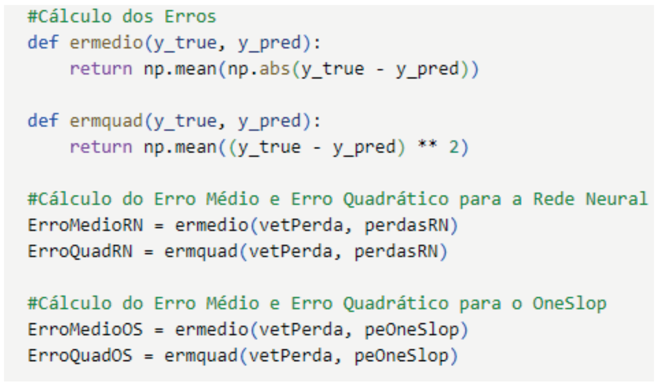


```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
The maximum number of train epochs is reached
Valores Medidos de Perda:
[-50. -48. -56. -55.]
Valores Calculados a partir da Rede de Perdas:
[-50.  -48.  -55.5 -55.5]
peOneSlop [-48.76165386 -49.14388397 -49.51140923 -49.86531926 -50.20658692
 -50.53608435 -50.85459647 -51.16283218 -51.46143394 -51.75098575
 -52.03202009 -52.30502377 -52.57044302 -52.82868789 -53.08013605
 -53.3251361  -53.56401051 -53.79705813 -54.02455646 -54.24676365
 -54.46392024 -54.67625073 -54.88396496 -55.08725941 -55.28631825
 -55.48131438 -55.67241032 -55.85975904 -56.04350467 -56.2237832
 -56.40072305 -56.57444563 -56.74506585 -56.91269258 -57.07742905
 -57.23937325 -57.39861824 -57.55525255 -57.70936038 -57.86102196
 -58.01031372]
```

Fig. 7 Results of models .

G. *Error evaluation*

In this step, the average errors and squared errors for the MLP-NRNA predictions were calculated in relation to the actual results and the values estimated by the OneSlope model. This evaluation quantified the performance of the MLP-NRNA in terms of its ability to accurately predict Wi-Fi signal losses.

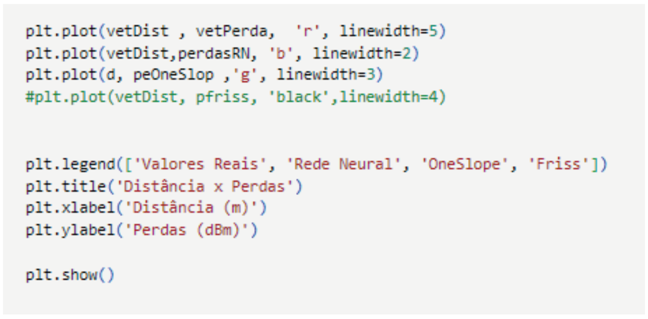


```
#Cálculo dos Erros
def ermedio(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred))

def ermquad(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

#Cálculo do Erro Médio e Erro Quadrático para a Rede Neural
ErroMedioRN = ermedio(vetPerda, perdasRN)
ErroQuadRN = ermquad(vetPerda, perdasRN)

#Cálculo do Erro Médio e Erro Quadrático para o OneSlop
ErroMedioOS = ermedio(vetPerda, peOneSlop)
ErroQuadOS = ermquad(vetPerda, peOneSlop)
```

Fig. 8 Calculation of the average error and squared error for the models used.

H. *Construction of graphs for data presentation*

In order to better visualise the models used, graphs were drawn in Python, using the matplotlib.pyplot library. This library has a wide variety of graphs and in this research the point graph was used, based on the distance and losses measured.



```
plt.plot(vetDist , vetPerda,  'r', linewidth=5)
plt.plot(vetDist,perdasRN, 'b', linewidth=2)
plt.plot(d, peOneSlop ,'g', linewidth=3)
#plt.plot(vetDist, pfriss, 'black',linewidth=4)


plt.legend(['Valores Reais', 'Rede Neural', 'OneSlope', 'Friss'])
plt.title('Distância x Perdas')
plt.xlabel('Distância (m)')
plt.ylabel('Perdas (dBm)')

plt.show()
```

Fig. 9 Code developed for creating graphs and presenting results.

V. NUMERICAL AND MEASUREMENT RESULTS

The test environment was Building 6 of the School of Engineering, which has three floors of classrooms. The measurements were carried out on the 5 GHz Wi-Fi signal.

The Table II contains information on 5G signal loss measurements carried out in different rooms, where the crucial input for the prediction is the distance between the router and the measurement point.

TABLE II – MEASUREMENTS TAKEN TO TRAIN THE NEURAL NETWORK

BUILDING 6/ 3rd FLOOR						
Environment	Heights (m)	Length (m)	Depth (m)	Distância Roteador – Ponto (m)	Losses measured in Notebook(dB)	Losses measured in Mobile (dB)
302	3.65	7.51	11.1	1	-31	-45
311	3.96	6.85	6.13	20	-87	-89
312	3.95	8.44	6.07	16.7	-89	-90
313/315	3.98	6.3	12.7	13.5	-86	-76
314	3.96	8.45	6.6	13	-85	-85
304	3.98	7.16	8.8	-	-	-
305	4.02	6.15	8.4	-	-	-
306	4.01	8.84	8.47	-	-	-
307	4.01	6.17	6.45	-	-	-
308	4.16	9.91	6.44	-	-	-
309	4.03	7.1	7.81	-	-	-

The measurement procedure was carried out as follows :

- Room: The number of the room or environment where the measurements were taken.
- Distance Router - Point: The distance in some standard of measurement (such as metres) between the router (signal source) and the measurement point inside the room.
- Notebook: The amount of signal loss in decibels when measured with a notebook.
- Mobile phone: The amount of signal loss in decibels when measured with a mobile phone.

As seen in the Table II the signal is no longer detected by the notebook's measurement software or by the mobile phone from room 314. since rooms 311. 312. 313/315 and 314 are geographically close to room 302. where the 5G signal transmitter is located. Now let's understand how this data can be used to train an Artificial Neural Network (ANN):

a) ANN input: The distance between the router and the measurement point (Router - Point Distance) will be used as the input for the ANN. This means that the ANN will learn the relationship between distance and signal loss.

b) Desired ANN output: The desired ANN output is the predicted signal loss. You can choose to use the measurements made with the notebook (Notebook) or with the mobile phone (Mobile) as the target output for training.

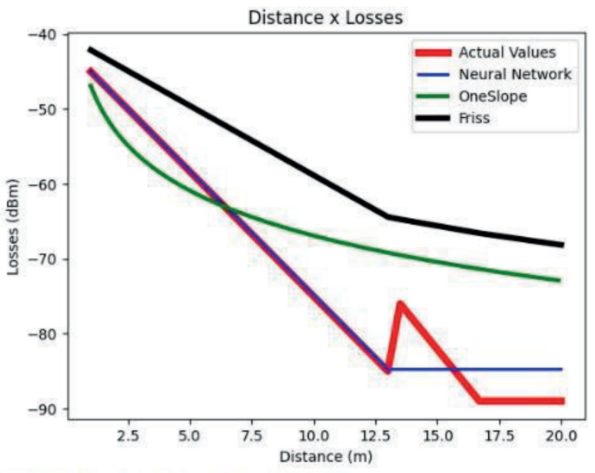
c) Data preparation: The distance will be normalised. making it compatible with the requirements of the ANN. This normalisation ensures that all the inputs are in the same range of values, which helps with training.

d) Creating the ANN: The ANN will be designed with an architecture that includes an input layer (corresponding to the distance), one or more hidden layers and an output layer.

e) Training the ANN: It was trained with the normalised distances as input and the signal losses (Notebook or Mobile) as target output. During training, the ANN will adjust its weights to minimise the error between the predictions and the actual values.

f) Evaluation and Adjustment: Performance is evaluated using error metrics such as the Mean Absolute Error (MAE) or the Mean Squared Error (MSE). calculated with the neural network predictions and the actual losses. Adjustments to the hyperparameters can be made based on these results.

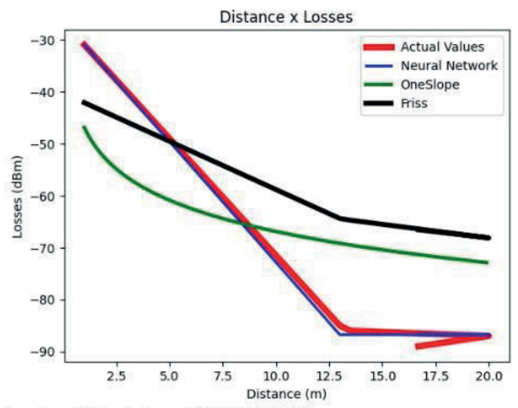
g) Using the trained neural network: After training, the neural network can be used to predict signal losses in new rooms based on the distances between the routers and the measurement points. This is useful for estimating the quality of the 5G signal at different distances.



```
Mean Neural Network Error: 3.50000002174927
Neural Network Quadratic Error: 22.549999999554664
OneSlop Average Error  : 19.478772908747395
OneSlop Quadratic Error: 440.97505160924834
```

Fig 10 . Losses measured on the 3rd floor.

The Figure 10 shows the values obtained from measurements made on a notebook and a mobile phone to obtain reference values for training the neural network at the same observation point. The neural network is being trained with the values presented and in relation to the reference model.



```
Mean Neural Network Error: 0.9999999483984411
Neural Network Quadratic Error: 1.7499998186282835
OneSlop Average Error  : 23.878772908747397
OneSlop Quadratic Error: 639.0599599742586
```

Fig 11 .   Losses measured on the 3rd floor.

The Figure 11 shows the values obtained and the convergence of the measurements for the trained neural network. There is a loss of reference for the most distant values.

VI. CONCLUSION

This work developed a tool based on the Python programming language, in the COLAB programming environment, to predict signals in wireless networks in indoor environments, using neural networks as a method of improving the accuracy of predictions. The main focus was on dealing with the challenges of signal propagation indoors, where obstacles, interference and attenuation can cause significant variations in wireless network signals.

The main objective of the project was to face these challenges by combining data obtained through detailed mapping of the indoor environment and real measurements of signal strength. The use of neural networks has demonstrated effectiveness in creating more accurate predictive models compared to traditional approaches. The developed Python tool allowed training and adjusting these models based on the collected information, considering the physical structure of the environment, present obstacles, and construction materials. The results obtained indicated significant improvements in the accuracy of forecasts compared to conventional methods, showing a great potential for the application of the tool in real-world scenarios. This includes planning and optimizing wireless networks in indoor environments, contributing to more stable and reliable connectivity. The project employed the methodology of creating and training an Artificial Neural Network to predict Wi-Fi signal losses, comparing the results of the neural network with field measurements and a traditional analytical model (OneSlope model). Comparison with real measurements and the analytical model revealed the ability of RNA-MLP to overcome the limitations of the traditional model and provide more accurate predictions, especially in scenarios with signal obstructions.

ACKNOWLEDGMENT

We would like to thank the MackPesquisa fund for supporting the scholarship and the research. We would also like to thank the engineering school for providing their laboratories and rooms for the tests. We would like to thank our colleagues at the U.P.M. Digital TV Laboratory for their support and access to the measurement equipment.

REFERENCES

[1] K. Pahlavan and A. H. Levesque, Wireless Information Networks 2th ed., Ed. Wiley, Chichester, England, 2005, page(s): 1 – 4.

[2] HAYKIN, Simon. Redes Neurais: Princípios e prática. Porto Alegre RS:Bookman, 2001.