FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Trace-based ns3-gym Reinforcement Learning Environment Framework for Wireless Networks

Gonçalo Regueiras dos Santos

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Lopes Campos

Co-Supervisors: Eduardo Nuno Almeida and Helder Martins Fontes

September 2, 2021

Trace-based ns3-gym Reinforcement Learning Environment Framework for Wireless Networks

Gonçalo Regueiras dos Santos

Mestrado Integrado em Engenharia Informática e Computação

September 2, 2021

Abstract

Aerial wireless networks composed of Unmanned Aerial Vehicles (UAVs) acting as aerial Wi-Fi Access Points (APs) or cellular base stations have come forth as a potential approach to wireless networks in a multitude of scenarios, such as emergency scenarios and crowded events. To ensure that the Quality of Service (QoS) requirements are satisfied, it is crucial to position the UAVs according to the users' traffic demand. Reinforcement Learning (RL) has emerged as an interesting solution to solve several wireless network problems such as this one. In RL algorithms, an agent learns on its own how to make optimal decisions based on experience gained in previous interactions with the environment.

A problem common to all wireless networking research and development is the validation phase. Validation is usually done through experimentation, in a real-world environment. This brings many challenges, including setup and operational costs, external interference, which, among other problems, might compromise the repeatability and reproducibility of an experiment. Simulation can be used as an easy-to-use estimation, but due to the use of models that simplify real phenomena, their results can vastly differ from real experiments. Merging these two approaches enables researchers to run simulations with real experimental results, while maintaining all simulation advantages, such as simplicity, cost, repeatability and reproducibility, all key factors when validating or evaluating a proposed wireless network solution. These trace-based simulation approaches rely on repeating real experiments conditions in simulation. However, they all share a major limitation, they only allow the repetition of the exact conditions of an experiment, not supporting any variation, such as the number of nodes, trajectory, or experiment duration.

The main goal of this work is to create a framework, based on a trace-based approach that supports developing, training and evaluating RL algorithms for wireless networks. This work provides an enhanced tool to improve network simulations and solutions, even when a real testbed is not available or cannot be used. We propose a novel trace-based approach, by training a machine learning (ML) model able to generate dynamic variations of a given scenario, which is characterised by the corresponding traces. The ML trace-based approach can then be used in the ns-3 network simulator, together with ns3-gym, a reinforcement learning framework, that enables the creation of an OpenAI Gym environment within ns-3. Additionally, this model can be used independently to run network simulations with improved accuracy.

ML trace-based propagation loss models showed to be more accurate than their classical counterparts in most of the scenarios. Simulations done in ns-3 showed that improved propagation loss models are useful, both when analysing simulation data from ns-3 and when training a RL agent using ns3-gym. This framework can be used to represent different scenarios, as long as new data is collected and the ML models are trained on it.

Keywords: Wireless Networking Experimentation, Trace-Based Simulation, Reinforcement Learning, Wireless Networks

Resumo

As redes aéreas sem fios compostas por veículos aéreos não tripulados (UAVs) atuando como Pontos de Acesso Wi-Fi aéreos (APs) ou estações base de rede móvel surgiram como uma potencial abordagem às redes sem fios numa multiplicidade de cenários, tais como cenários de emergência e eventos com multidões. Para assegurar os requisitos de Qualidade de Serviço (QoS), é crucial posicionar os UAV de acordo com a necessidade de tráfego dos utilizadores.Aprendizagem por Reforço (RL) surgiu como uma solução interessante para resolver vários problemas de redes sem fios, tais como o posicionamento dos UAVs. Nos algoritmos de RL, um agente aprende como tomar decisões óptimas com base na experiência adquirida em interacções anteriores com o ambiente.

Um problema comum a toda a investigação e desenvolvimento de redes sem fios é a fase de validação. Esta é geralmente feita através de experimentação, num ambiente real. Isto traz muitos desafios, incluindo custos de instalação e operacionais, interferências externas, que, entre outros problemas, podem comprometer a repetibilidade e reprodutibilidade de uma experiência. A simulação pode ser usada como uma estimativa fácil de usar, mas, devido à utilização de modelos que simplificam fenómenos reais, os seus resultados podem ser não corresponder aos reais. A fusão destas duas abordagens permite aos investigadores executar simulações com resultados experimentais reais, mantendo todas as vantagens da simulação, tais como simplicidade, custo, repetibilidade e reprodutibilidade, todos factores-chave ao validar ou avaliar uma solução de rede sem fios. Estas abordagens de simulação baseadas em traços dependem da repetição das condições reais das experiências na simulação, no entanto, todas elas partilham uma limitação. Apenas permitem a repetição das condições exactas de uma experiência, não suportando qualquer variação, como o número de nós, trajectória, ou duração da experiência.O objectivo principal deste trabalho é a criação de uma solução, baseada em traços, que suporte o desnvolvimento, treino e avaliação de algoritmos de RL no contexto de redes sem fios. Este trabalho fornece uma ferramenta para melhorar as simulações de redes sem fios para quando a utilização de um banco de ensaio não é possível. É proposta uma nova abordagem baseada em traços, através do treino de um modelo de aprendizagem de máquinas (ML) capaz de gerar variações dinâmicas de um determinado cenário, caracterizado pelos traços correspondentes. A abordagem baseada em ML pode ser utilizada no simulador de redes ns-3, juntamente com o ns3-gym, uma infraestrutura de RL, permitindo a criação de um ambiente OpenAI Gym em ns-3. Além disso, este modelo pode ser utilizado independentemente para executar simulações de rede com maior precisão.Os modelos de propagação baseados em ML e traços mostraram-se mais precisos que os seus equivalentes determinísticos na maioria dos cenários. Simulações realizadas no ns-3 tiraram partido da melhoria nos modelos de propagação, quer analisando os dados do ns-3 quer quando são utilizados para o treino de agentes de RL através do ns3-gym. Esta solução pode ser utilizada para simular inúmeros cenários e ambientes desde que novos dados nesses ambientes sejam recolhidos ou utilizados e que os modelos de ML baseados em traços sejam treinados com esses novos dados.

Acknowledgements

I would first like to express my gratitude to my supervisors, Prof. Rui Campos, Eng. Eduardo Almeida and PhD Helder Fontes, who accompanied me since the beginning of this dissertation, with great wisdom and who were always available to explain and help me understand more about wireless networks.

I am deeply grateful to all the teaching and supporting staff at FEUP, for all the academic, technological and personal growth experiences, that through many challenges helped me grow and become the person that I am today.

Finally, I can not thank my family and friends enough, for putting through all my grumpiness and supporting me throughout this five year journey. A very special thanks to my sisters, who helped me transforming my thoughts in words, even when they were a chaotic theory on my head that even I did not fully understand.

This article is a result of the project "DECARBONIZE – DEvelopment of strategies and policies based on energy and non-energy applications towards CARBON neutrality via digitalization for citIZEns and society" (NORTE-01-0145-FEDER-000065), supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

Gonçalo Regueiras dos Santos

"I am in a charming state of confusion."

Ada Lovelace

Contents

List of Figures

List of Tables

Abbreviations

AP	Access Point
PR AUC	Area Under Precision-Recall Curve
ROC AUC	Area Under Receiver Operating Characteristic Curve
ANN	Artificial Neural Network
CWI	COST-Walfisch-Ikegami
DL	Deep Learning
DQN	Deep Q-network
DRL	Deep Reinforcement Learning
DDQN	Duelling Deep Q-network
GPSR	Greedy Perimeter Stateless Routing
ML	Machine Learning
MC	Monte Carlo
MLP-NN	Multilayer Perceptron Neural Network
OSPF	Open Shortest Path First
OLSR	Optimised Link State Routing
PL	Path Loss
QoS	Quality of Service
RBF-NN	Radial Basis Function Neural Network
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RMS	Root Mean Squared
SVM	Support Vector Machine
SVR	Support Vector Regression
TD	Temporal-difference
T-OLSR	Trajectory Optimised Link State Routing
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Context

Wireless networks are a key component of modern life as we know it, and they are a hot topic of research [52, 30]. They can be composed of one or more Access Points (APs) that communicate with several kinds of devices, allowing them to communicate with each other without the need for a wired connection. Recently, aerial wireless networks have appeared as a promising solution for dynamic scenarios, due to their versatility and applicability in many domains, from crowded events to emergencies, that traditional networks may not be able to handle. These networks are partially or entirely composed of flying nodes – Unmanned Aerial Vehicles (UAVs) – serving as Wi-Fi APs or cellular base stations. To fulfil the Quality of Service (QoS) requirements, the UAVs' position must be chosen according to the users' traffic demand.

Reinforcement Learning (RL) has emerged as an interesting solution to multiple network problems [50, 20], such as the UAV placement problem in flying networks [29], where each UAV acts as an autonomous training agent that learns how to position itself, guaranteeing acceptable QoS levels, based on experience previously collected in interactions with the environment.

To develop and evaluate an RL algorithm's performance, an RL environment must be created, where an agent can interact and observe the conditions of the environment [24]. The ns3-gym [17] framework enables the creation of RL network environments, leveraging the interface provided by OpenAI Gym [6] and the network simulation provided by ns-3[1]. A standard OpenAI Gym interface is defined through this framework to access the observations, actions, and rewards provided by an underlying ns-3 network simulation.

To assist wireless networks development efforts, several network simulators were created – e.g., ns-3 and NetSim[2]. Originally, these simulators have theoretical equations and models as a basis. However, this comes with some disadvantages, as those models cannot accurately reproduce all characteristics of a real environment. To better approximate simulators to real-world conditions, several researchers have proposed mixed approaches, where data collected from real experiments

[1]http://www.nsnam.org/

[2]https://www.tetcos.com/

– network traces – is integrated into the network simulators and complements theoretical models, creating trace-based simulations [1, 13, 35].

Trace-based simulations offer most of the advantages of computer simulations, such as ease of reproducibility and repeatability, lower costs, and improved accuracy compared to more classical approaches. Nonetheless, trace-based simulations do not offer that much flexibility, as the network topology has to follow the one registered in the experiment.

1.2 Motivation

As real experiments have a complex and costly setup, simulations can be helpful. Experiments in testbeds are subject to testbed availability or reproduction conditions, leading to an increased difficulty to repeat and reproduce other experiments. This hinders the capacity of researchers validating the results of others, a key component of the scientific method. Some trace-based approaches have been proposed to eliminate this problem, allowing the repetition of the experiment conditions in a simulation – either at the physical level [13, 14] or at the application level [1, 35]. However, all these approaches share a common flaw, only supporting complete repetition of the experiment, not supporting any kind of change to it, such as changing the trajectory of a node or increasing the duration of the experiment. As such, an approach that allows the dynamic generation of scenarios based on an experiment is a valuable contribution, improving the accuracy of simulations and reducing the need for experimentation in testbeds. Such enhanced simulations could then be used in several domains, such as the training environment of an RL agent.

1.3 Objectives

The major objective of this work is to develop a framework based on a trace-based approach that supports developing, training and evaluating RL algorithms for wireless networks. To accomplish this, the major objective can be divided into three sub-objectives:

- Creation of a Machine Learning (ML) trace-based propagation loss model able to dynamically generate scenarios similar to the one characterised by the collected traces;
- Integration of the ML trace-based model with ns-3 and ns3-gym, creating a trace-based ns3-gym framework. This framework will enable the training and evaluation of RL algorithms for wireless networks;
- Validation of the trace-based ns3-gym framework performance by comparing it with an agent trained in ns3-gym using a pure simulation[3] approach.

1.4 Original Contributions

At the end of this dissertation, the original contributions are:

[3]Pure simulations are simulations based on not trace-based models, such as Friis or Log-distance.

1. **ML trace-based propagation loss model.** A ML model able to learn both the primary and the fast-fading component of the path loss from traces collected in an environment. This strategy allows the model to replicate the conditions of the environment, not only repeating the collected traces but rather be the basis for the creation of new ones in a virtual representation of the original environment.

2. **Trace-based ns3-gym framework.** An ns3-gym framework, where users can leverage the earlier mentioned models, combining the capabilities of the ns-3 simulator and the easy to use interface of ns3-gym to train and develop new RL algorithms and environments.

1.5 Document Structure

The remainder of this document is structured as follows. Chapter 2 focuses on literature review on machine learning concepts, state-of-the-art supervised and reinforcement learning algorithms, methods of calculating signal propagation loss and existing trace-based simulation approaches and related work in these areas. Chapter 3 defines the problem, proposes a solution and explains its implementation. Chapter 4 analyses the solution performance and behaviour in multiple test scenarios. Finally, Chapter 5 concludes the document and identifies the future work.

Chapter 2

Literature Review

This chapter presents an analysis of the state-of-the-art and related work for the main topics of this dissertation. It is divided into four sections: machine learning, focused on fundamental machine learning concepts; machine learning models, where some supervised and reinforcement learning algorithms are introduced; path loss models, discussing both theoretical and machine learning approaches to model path loss; network simulation based on traces of experiments; and conclusion, linking all these topics.

2.1 Machine Learning

Machine Learning is a branch of Artificial Intelligence responsible for identifying patterns in data and autonomously making decisions on it, improving its performance by learning from previous samples. As Bishop [5] defines it, “pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories”. This concept can be extended to other domains, such as predicting the value of a continuous variable. Usually, a machine learning model is created by feeding a machine learning algorithm a set of data which is used to train a model to make predictions on new data items. The algorithm is expected to deduce the correlation between the training data features’ values and the target variables.

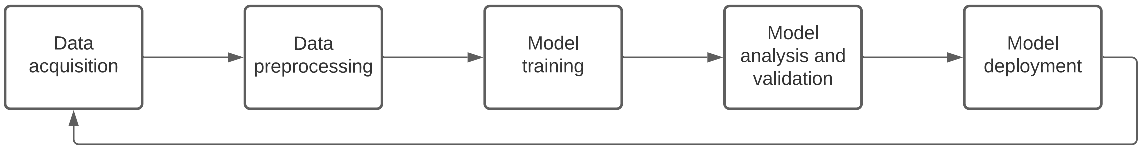


Figure 2.1: Machine Learning workflow.

As can be observed in Figure 2.1, the workflow in machine learning passes through multiple steps, from acquiring the data to deploying the model in production, and it is cyclic, as a ML model requires maintenance, to make sure it stays relevant and accurate.

Ideally, one of the first decisions that is made in ML problems is the definition of the evaluation method and success metrics. Commonly used evaluation methods are accuracy, percentage of correct prediction out of all predictions; precision, percentage of positive instances out of all predicted positives; recall, percentage of positive instances out of the total actual positives; area under precision-recall curve (PR AUC), a graph plotted against the precision and recall values for different threshold values and area under receiver operating characteristic curve (ROC AUC), a graph plotted against true positive ratio and false positive ratio for various threshold values. Both accuracy, precision and recall are too simplistic and, depending on the problem, it is possible to achieve high values of them with a completely useless model. ROC AUC and PR AUC are more robust measures, as they provide a clearer picture of the model's performance. *Saito et al.* [41] strongly recommend that PR AUC is a more representative metric than ROC AUC in tasks with unbalanced data, where the ratio between samples from the negative class outweigh positive class ones.

After acquiring the data and before passing the training data to the model, most of the times, the input values are transformed to simplify the pattern recognition problem or reduce the computation cost, in a preprocessing step. When preprocessing is applied to the training data, it must also be applied to the test data, as the model does not know how to infer knowledge based on the real data, but only on the transformed version of it. New features can be created, inferred from others, to highlight some patterns that may be easy for humans to infer, but difficult to machines, in a process called feature engineering. While this feature extraction from raw data process can be manually done by humans, recently Deep Learning (DL) has proved to be an interesting solution to automate it [31]. DL is an ML approach that uses multiple neural network layers, mimicking the human brain structure, to extract high-level features from complex raw data.

After training a model, it can be used on new samples, usually with low computational costs, and then evaluated based on the defined metrics. This whole process can be repeated several times – e.g., modifying the model hyper-parameters or using a different preprocessing technique – until reaching satisfying values.

Machine Learning problems can be divided into three major categories: supervised, unsupervised learning and reinforcement learning. In supervised learning problems the true value of the target variables in the training data are known, while in unsupervised those values are unknown. Reinforcement learning problems are characterised by an interaction with the environment, as the agent learns by *trial-and-error* from feedback provided by the environment. Following the same taxonomy applied to machine learning problems, machine learning models can also be divided in three main groups: unsupervised, supervised and reinforcement learning models. As the former is out of scope for this dissertation, it won't be covered in detail.

ML has been used with success to solve several networking problems, such as cache management [46] where reinforcement learning was used to design an efficient caching strategy, outperforming random caching and caching based on time-averaged content popularity. Supervised learning has also been adopted to solve network routing problems, such as [25] where deep learning was used to route packets through a wireless network, maximising the total throughput and achieving better results than state-of-the-art routing strategies such as open shortest path first (OSPF).

When compared to traditional methods, ML-based methods have several advantages [45] including creating algorithms with similar performance to traditional algorithms but with a reduced complexity, leading to lower computational expense; successfully lead with lack of network information and knowledge gaps, as some traditional methods that require global knowledge of the network can be replaced with reinforcement learning associated to transfer learning – where a model trained in a scenario is used to bootstrap the training in other scenario – to minimise the energy spent in cellular radio access networks [28], achieving results almost as good as state-of-the-art algorithms that require complete knowledge of traffic loads in all nodes; easier self-organisation capabilities [3]; and in some cases, better performance than traditional optimisation methods [49].

2.1.1 Supervised Learning

Supervised learning problems can be further classified, regarding their goal, as classification and regression problems. Classification problems try to assign each input data to one of a finite number of classes or categories – e.g., identifying a handwritten digit. Regression tasks try to predict the value of one or more continuous variables – e.g., forecasting houses price. Supervised learning is commonly applied in the wireless network context to solve multiple problems [48], such as signal strength prediction problems [53]. Section 2.2 details several supervised learning algorithms.

2.1.2 Unsupervised Learning

Unsupervised learning tasks are characterised by the lack of a target value for each sample. The absence of a true ground value, brings some challenges on how to evaluate those systems, as a hard metric is sometimes difficult to reach. Usually, it can be divided in two groups: Clustering [22] – create groups of similar data – and Association Rules [19] – determine interesting relationships or dependencies in large sets of data items. Unsupervised Learning is many times used as a complement for supervised learning problems, e.g., annotating large datasets or to help gain some insight into how the data is structured before developing a supervised learning approach. When clustering a dataset, a commonly used algorithm is K-means [23]. K-means was first published in 1955, but still is one of the most popular clustering algorithm [23]. It works by choosing K points as cluster centres and assigning each data point to its closest cluster centre. Afterwards new cluster centres are computed and the data points are reassigned. This step is repeated until cluster membership stabilises.

Agrawal et al. [2] proposed an algorithm to find association rules in data. It is based on the *Apriori property* that if an item set is frequent, then all its non-empty subsets are also frequent. It starts by computing the frequency (support) of each item and discarding the ones whose support is inferior to a threshold. In the second iteration, the support of all item sets of size two formed from pairs of single items that survived the first pass is computed and pairs with support inferior to the threshold are removed. In the next iterations, the item sets that survived the previous iteration are combined with those retained from the first iteration, discarding those with support inferior to the threshold. This is repeated until no more combinations can be made.

Unsupervised learning can be used in a myriad of scenarios in the wireless network context, such as detecting intrusions. *Zhong et al.* [42] applied clustering techniques to detect intrusions and anomalous behaviour in wireless networks.

2.1.3 Reinforcement Learning

A reinforcement learning agent learns by sequentially interacting with a dynamic environment and trying to maximise the obtained rewards. *Kaelbling et al.* [24] defines a reinforcement learning model as a set of states, a set of possible actions and a set of scalar rewards. At each step, the agent receives as input the environment's state; chooses an action, changing the environment's state; and receives a reward, as Figure 2.2 depicts. The agent's main task is to infer a policy that maps states to actions and maximises the expected total reward.

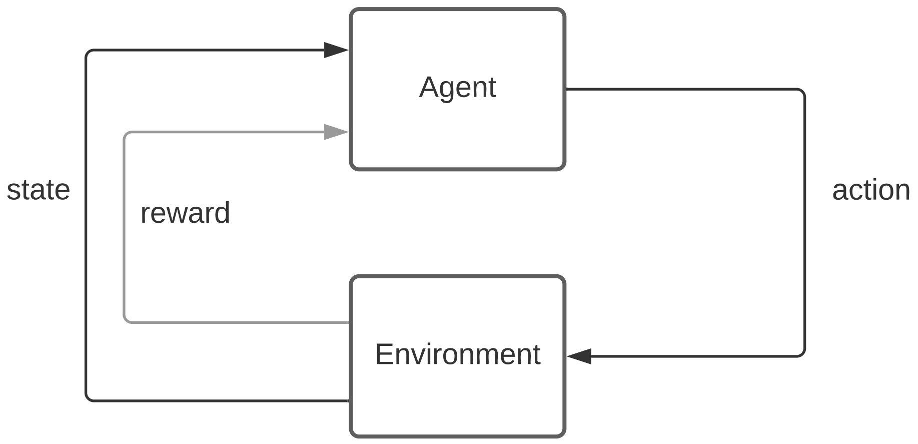


Figure 2.2: Reinforcement Learning algorithm.

Multiple algorithms have been proposed for training the agent's policy. Those algorithms can be divided in two major groups: Monte Carlo and Temporal-difference. Both are *model-free* in the sense that they do not need to know all the states and possible actions, as they learn from episodes of experience. Recently, deep reinforcement learning (DRL) [32] has appeared as an interesting solution to solve several problems in different domains [32, 34], including wireless networks. DRL merges the self-learning capacity of RL with the perception capacity of DL solutions, creating RL agents able to extract deeper features from the networks which can help to speed up the learning phase and achieve better results. DRL can autonomously infer the environment's state from its observation instead of relying on a human-designed set of states, transforming the observation space from discrete to continuous.

Monte Carlo (MC) [7] methods are present in many areas of computing and mathematics. In general, all these methods work by generating a set of random numbers or experiences and then inferring some kind of knowledge based on a fraction of the numbers obeying some property or group of properties. When applied to reinforcement learning, it works by running multiple episodes until the end, and averaging the returns for the visited states in each episode. Two main strategies exist, *First-visit MC* and *Every-visit MC*. In the first, only the first time each state is visited is included in the average calculations while in the latter it is included every time it is visited.

Temporal-difference (TD) [10] approaches are characterised by continuously updating their policy. TD approaches share some similarities with MC ones, in the sense that they learn from samples of the environment but they differ in the policy updating strategy, as they update their state-value prediction as they explore the environment, based on the current estimate of the next state and the reward obtained along the way. The value of a state is iteratively updated following the equation depicted in Equation (2.1), where $V(s)$ and $V(s')$ are the value estimate for the old and new states, respectively; α is a learning factor; r is the observed reward; and γ is a discount factor.

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s)) \tag{2.1}$$

Reinforcement Learning has proved to be highly helpful in dealing with multiple problems, in many different domains, such as flying and non-flying wireless networks.

Hou et al. [21] uses Q-learning, a TD algorithm, to improve the *optimized link state routing* (OLSR) protocol, creating a novel trajectory-OLSR (T-OLSR) algorithm, applied to sparse UAV networks. Compared to the state of the art algorithms, such as *greedy perimeter stateless routing* (GPSR) and the original OLSR, the packet delivery ratio improved by over 30%, reducing the end-to-end delay by more than 40s in some conditions.

Nie et al. [34] proposes a deep reinforcement learning approach to solve the energy efficiency problem in a wireless flying communications network. In this situation, fixed ground devices are powered by the wireless signal of a flying rotary-wing UAV. As such, the goal is to design the UAV trajectory, minimising the energy spent and guaranteeing acceptable levels of QoS to the ground devices. The UAV acts as an agent, receiving observations and a reward from the environment. For computational simplification, UAV's height is fixed, its speed is constant, and its actions are moving horizontally into one of eight directions or not moving. To learn the best policy for the agent's behaviour, it uses a deep Q-network (DQN) algorithm, which combines Q-Learning with deep learning techniques. An artificial neural network is used to estimate the action-value function. Furthermore, a duelling DQN (DDQN) is adopted for action selection to improve the convergence speed. An iterative approximation was considered the optimal value to evaluate the proposed framework, and a random approach was introduced as the baseline. Both DQN and DDQN approaches achieved similar average energy efficiency results per episode, with DDQN having a slight advantage over DQN. DQN and DDQN accomplished better results than the random approach but were marginally worse than the iterative one. The obtained results indicate that reinforcement learning, associated with deep neural networks, can potentially be used to solve flying network problems.

2.2 Supervised Learning Models

All supervised learning models share a common goal: learn from a labelled dataset and apply it in either classification or regression problems. In what follows, we refer to a set of supervised learning models available in the state of the art.

2.2.1 Decision Trees

Decision Trees are one of the simplest machine learning models. The input's prediction is modelled as a path, starting from the tree's root and ending on a leaf. A linear decision is taken in each node regarding the value of one of the input variables, choosing a new branch, until reaching a terminal node – a leaf – with a target variable value associated. Their main advantages are the relative simplicity of the model, allowing some interpretability and resilience to overfitting. Initially, this strategy was designed to solve classification problems. However, it can be adapted for regression problems with some limitations. Decision trees cannot fully model the output variable's domain, as the values at the leaves are restricted to a finite set of possible output values.

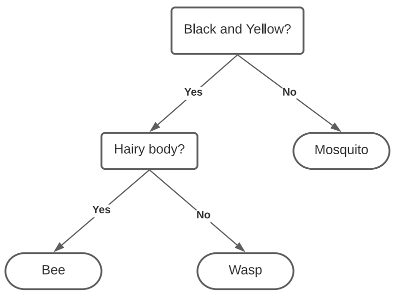


Figure 2.3: Example of a decision tree.

Figure 2.3 depicts the task of classifying a flying insect animal like a bee, a wasp or a mosquito. The two input variables relevant to the classification task are:

- if the insect is black and yellow or not;
- if the insect has a hairy body or not.

The three possible classes are *wasp*, *bee* and *mosquito*. This model can be easily interpreted, and from it, it is possible to infer three rules:

- if the insect is not black and yellow, it is a mosquito;
- if the insect is black and yellow and has a hairy body, it is a bee;
- if the insect is black and yellow and does not have a hairy body, it is a wasp.

This set of rules are easy to understand, for both machines and humans, and can classify any new sample by following a path from the tree's root to one of its leaves.

2.2.2 Random Forests

Random forests take advantage of Decision Trees' characteristics, combining several decision trees, trained on different samples of the training set. They are especially suited for data sets where the number of features is large, as they randomly select a subset of those features for each tree. Their main advantages are the relative level of interpretability and predictive performance. As more trees are used, the variability of the results lowers, while the computational expense increases. *Moreira et al.* [33] indicates that the recommended number of trees varies between 1000 and 5000, and the number of attributes to be used at each split-node is similar to the square root of the number of input attributes.

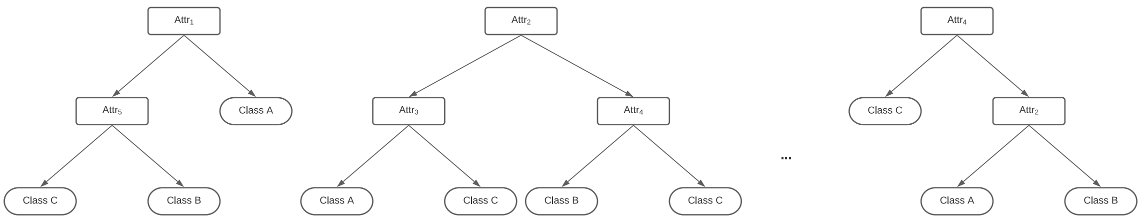


Figure 2.4: Example of a random forest.

Figure 2.4 depicts an example of a random forest, for a classification task with three output classes and five input attributes. When a new sample is processed, it passes through all decision trees and their result is aggregated into a final prediction.

2.2.3 Boosting

Boosting is a Machine Learning strategy where several weak learners – learners slightly better than random guesses – are combined into a strong model. Several algorithms benefit from this approach, such as *Adaboost* [15], *XGBoost* [9] and *Light GBM* [26].

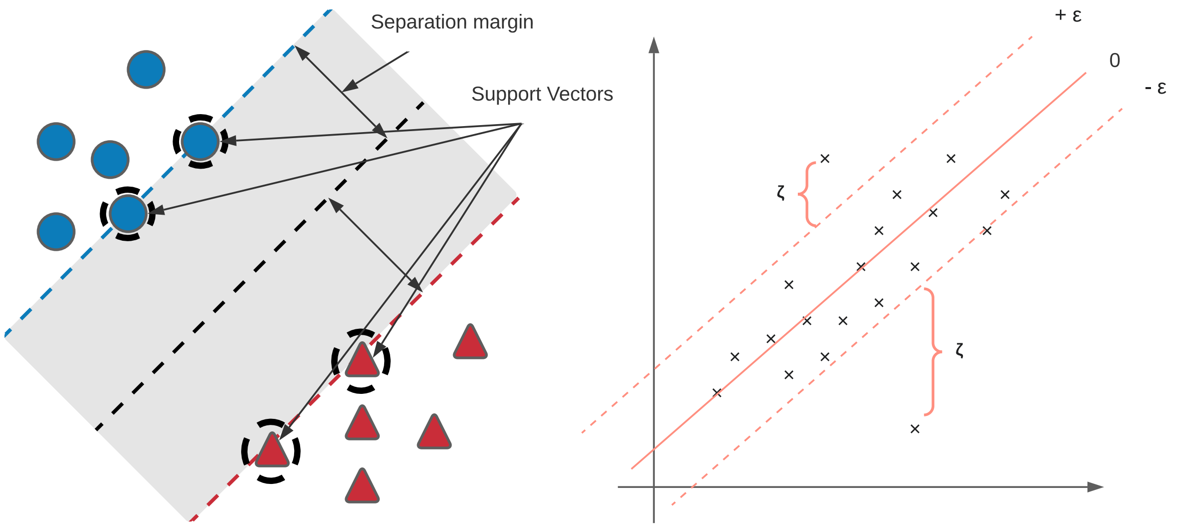
Adaboost works by sequentially training decision trees as weak learners, assigning larger weights to misclassified samples. After training each learner, it is assigned a weight proportional to accuracy. After the training phase, the final prediction is made by a weighted majority vote for classification problems or a weighted median in case of regression problems. *Adaboost* has the advantage of having fewer hyperparameters that need to be tuned but is more sensitive to noisy data.

While *Adaboost* was initially designed to work as a classification technique, it can be adapted to work in regression problems. One of the most popular adaptations is gradient boosting, from where *XGBoost* is based. *XGBoost* stands for "extreme gradient boosting" and is especially suited for large quantities of data, as it is takes advantage of some optimisations, such as cache access patterns, data compression and sharding to improve its computational efficiency [9]. *LightGBM* is another approach to gradient boosting decision trees, applying novel techniques for sampling the data and reducing the number of features, resulting in up to 20 times faster training times while achieving similar accuracy to other methods [9].

2.2.4 Support Vector Machine (SVM)

Support Vector Machines (SVMs) can be applied to both regression and classification tasks. In binary classification tasks, the SVM learning algorithm works by selecting some samples as support vectors, that will be used to create a border and a separation margin ε, as can be seen in Figure 2.5a. While originally SVMs could only deal with linearly separable tasks, using kernel functions it is possible to transform non-linearly separable data into a higher-dimensional space, where it is linearly separable. When applied to regression tasks, the main difference is the error formula, where instead of maximising the separation margin, the goal is to minimise it and the error ζ is the sum of all samples outside that margin.

Figure 2.5b depicts a regression task, where a user defined ε margin is set and a soft margin 0 is found, minimising the ζ sum.



(a) SVM classification example. (b) SVM regressor example.

Figure 2.5: SVMs can be applied to both classification and regression tasks.

2.2.5 Artificial Neural Networks

Interest in artificial neural network (ANN) dates back to the early 1940's [47]; however, it was highly limited by the computational power available in that period. Recently they have been used in many different domains, with highly satisfying results. These networks bear some resemblance to how biological brains work, by having multiple *neurons* interconnected in layers.

As can be observed in Figure 2.6, an ANN can be split into three parts: an input layer, one or more hidden layers and an output layer. The input has as many nodes as the number of data features, and the output layer can have as many nodes as possible classes, in a classification task, or just one node, in regression tasks. The hidden layers have a much more diverse structure depending on the particular task at hands.

Each *node* (neuron) is connected to other nodes by *links* and its output is calculated by applying a mathematical function to the neuron's input. The output value of a neuron is fed-forward to the neurons in the following layer, according to the neural network's connection graph. Each link has a *weight* associated with it. This computation can be split into two: a *linear* computation, computing the weighted sum of the input links and a *nonlinear* computation, transforming the previously computed value with an *activation function* [40]. This activation function's behaviour can be seen as deciding whether the neuron is activated ("fired") or not, based on the neuron's relevance for the model's behaviour. It can be as simple as a step function, turning the neuron on and off, following some threshold, or more complex, mapping the input signals domain into a different domain. Some examples of these more complex functions are *Sigmoid* and Rectified Linear Unit (*ReLU*).

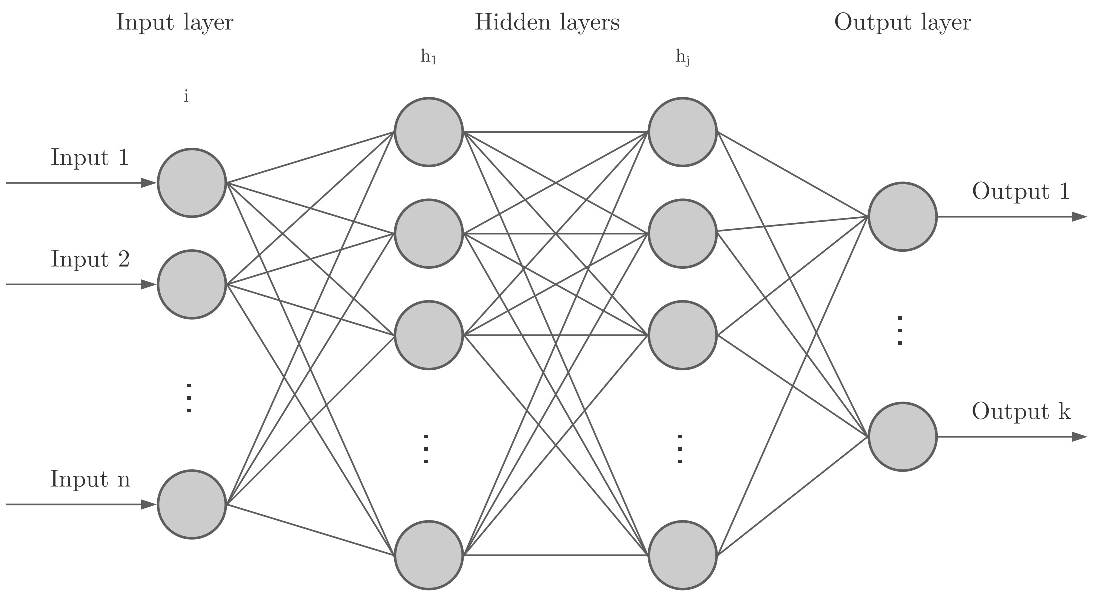


Figure 2.6: Artificial Neural Network example.

The training phase consists of learning the optimal weights in order to minimise the error between the ANN's predictions and the expected output, which is provided in the dataset. Labelled training data is supplied to the network that feeds it forward through all layers until reaching the output layer, where the error is calculated and propagated back, updating the associated link weights, until the error is smaller than a defined threshold or the network can't improve more. This process is known as back-propagation and was introduced in 1969 by *Bryson and Ho* [40]. The idea is that each node has a certain contribute to the error of its output nodes and that by distributing the error by the involved nodes, proportionally to the strength of their connection, we can tune those weights to reach an acceptable error level.

2.3 Path Loss Models

When developing or researching wireless networks systems, radio wave propagation is a key factor to take in consideration. As the quality of the network link highly depends on the received signal's strength, its modelling is a hot topic of research [37]. Throughout the years, several methods were proposed to model the path loss, defined as the signal strength lost throughout its path. They can be classified as theoretical, such as Friis [16] and Two-Ray [37]; empirical, for instance Log-distance [11] and ML-based, explained in more detail in Section 2.3.4. All approaches have some drawbacks, as classical theoretical models are too optimist since they do not take into account the specific phenomena and conditions of the environment and both empirical and ML-based models require a previous characterisation of the scenario, making them only suitable for that specific scenario.

2.3.1 Friis Model

The Friis model is a simple model which characterises the decreasing of the signal strength caused by distance. It is used to calculate the received power P_r, taking as parameters the transmission power P_t; the gain of the receiving and transmitting antennas, G_r and G_t, respectively; the signal's wavelength λ; and the distance between antennas d, as can be seen in Equation (2.2). Friis's original formula in 1946 [16] took the antenna's area into account. Nowadays, the area of the antennas is not considered [37], as they are considered ideal isotropic antennas.

$$P_r = P_t G_r G_t \left(\frac{\lambda}{4\pi d} \right)^2 \tag{2.2}$$

2.3.2 Log-distance Model

The Log-distance model considers that the path loss varies exponentially with distance, taking into account the shadowing that occurs. The path loss value L is defined by a reference loss value near the transmitting antenna L_0; an empirical path loss exponent γ, which is based on empirical observations; and the distance between antennas d, given by Equation (2.3). A Gaussian random variable X_g with zero mean can be added to this value, representing the attenuation caused by fading [4].

$$L = L_0 + 10\gamma \log_{10} \left(\frac{d}{d_0} \right) + X_g \tag{2.3}$$

2.3.3 Two-Ray Model

The two-ray model considers the effect of the signal's reflection, which can be positive or negative to calculate the path loss. It divides the signal into two components: a direct line of sight component d_{los} and a reflected one d_{ref}, as is represented in Figure 2.7.

As the complete formula that takes into account the phase difference φ between the components and the reflection coefficient Γ results in a too complex and computationally expensive

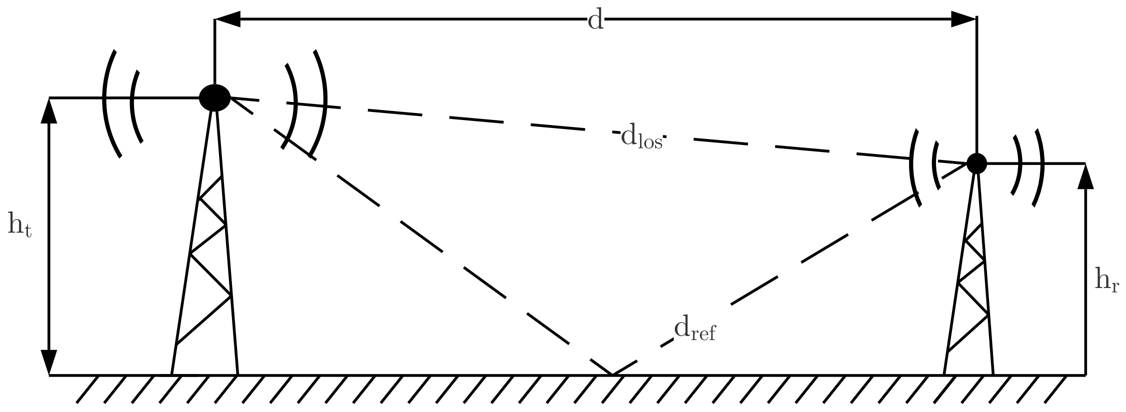


Figure 2.7: Two-Ray model diagram.

calculation – as can be seen in Equation (2.5) –, it is commonly simplified using the Friis' equation, if the distance d is inferior to a cross-over distance d_c – calculated by Equation (2.4) –, or by a formula assuming perfect polarisation and reflection if superior [52], as it is depicted in Equation (2.6). The gain of the receiving and transmitting antennas are represented by G_r and G_t, respectively.

$$d_c = \frac{4\pi h_t h_r}{\lambda} \tag{2.4}$$

$$Pr = 20\log_{10}\left(4\pi\frac{d}{\lambda}\left|1+\Gamma_{\perp}e^{i\varphi}\right|^{-1}\right) \tag{2.5}$$

$$P_r = \begin{cases} P_t G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 & \text{if } d \leq d_c, \\ P_t G_t G_r \left(\frac{h_t h_r}{d^2}\right)^2 & \text{if } d > d_c. \end{cases} \tag{2.6}$$

2.3.4 Machine Learning-based Models

As mentioned above, classical theoretical and classical empirical models are not perfect, as they lack accuracy and the empirical models have parameters that need to be manually adjusted to each scenario. Several techniques for estimating path loss using machine learning were developed to solve this problem. As ML-based models are trained with data collected from the location in study, its accuracy tends to be better than classical empirical models. Nonetheless, ML-based models suffer from increased complexity, due to the ML model development and training phases; data collection; and, as classical empirical models, lack of applicability in different scenarios.

Zhang et al. [53] makes a comparison between the collected data in an urban scenario in Beijing, China and the predicted values of path loss using ANNs, SVMs, Random Forests and the classical empirical log-distance model. Data was collected by a moving vehicle communicating with a fixed based station, at first with the signal obstructed by buildings and trees – out of line of sight – and then on a highway – within line of sight. The provided inputs were the distance between antennas and the used frequency. A test with synthetic data, generated by the log-distance model, added to the real collected data, performed well in a new frequency not present in the training data, having a root mean square error (*RMSE*) value between 1.61 and 2.52 dB. Adding 30 samples (from the new frequency set of 110 samples), proved to improve the models' performance, lowering the overall *RMSE* to values between 1.37 and 1.72 dB. The researchers' main challenges were the lack of enough data, both in quality and in number.

Cabral et al. [8] used the data collected in Huelva, Spain in an open environment scenario, where a UAV flew in the vicinity of a fixed base station. The selected features feeding the ML model were the UAV position – latitude, longitude and altitude – and its attitude – roll, pitch and yaw. At first, the direct received signal strength (RSS) prediction was the output of the models. Then, another approach was explored. Instead of having the machine learning models predict the received signal strength, they were provided with Friis model's prediction and trained to predict its difference to the theoretical Friis value, effectively correcting it to achieve the real result. Both approaches were benchmarked against Friis, Log-distance and Two-Ray models. In two of the used techniques – Adaboost and SVM – the direct RSS approach had a slightly lower mean absolute error than the Friis correction one, 2.59 dB to 2.91 dB and 2.52 dB to 2.69 dB respectively. In the other technique – Random Forest –, the Friis correction model showed improved performance compared to direct RSS prediction, 2.82 dB compared to 3.12 dB.

Popescu et al. have studied the usage of ANN based models both in indoor [38] as in outdoor [39] settings. To study the path loss estimation indoors tests were conducted in a university building, in a hall of offices, and compared with a modified Log-distance model, which took into consideration the number of walls penetrated by the signal. Two ANN topologies were considered, a Generalised Radial Basis Function Neural Network (RBF-NN) and a Multilayer Perceptron Neural Network (MLP-NN). Both achieved similar root mean squared (RMS) results, with RBF-NN improving MPL-NN's result by 0.15 dB. One of the ANN based model, not specified, improves the RMS value by 4.37 dB. To explore those ANN topologies in outdoor settings, the results achieved were compared to a COST-Walfisch-Ikegami (CWI) model. A hybrid prediction model was also considered, where the CWI prediction is used as the ground value of the models' output. Both ANN models achieved better results than CWI for both urban, in and out of line-of-sight, and suburban, out of line-of-sight, environments when directly predicting the path loss value. Analysing the obtained results, *Popescu et al.* concluded that both ANN studied and both prediction strategies have similar results and perform better than the CWI model.

2.4 Trace-based Simulation

As simulation is a crucial step in researching, developing, and validating new networking solutions, several researchers have been trying to create new solutions to improve network simulation accuracy. These allow better representations of real-world conditions and save time and resources, which are often limited. The central concept is common to several of these approaches, replacing some part of the network stack, previously approximated using some theoretical model, with real data collected from experiments. All the methods that we will explore were proposed for the ns-3 network simulator, which is widely used in the scientific community.

The approaches of *Agrawal and Vutukuru* [1] and *Owezarski Larrieu* [35] rely on replacing the application layer with a replay of previously collected network traces. The main advantage of this strategy is the increased accuracy compared to other ns-3 application layer models; however, it comes at the price of lower flexibility, as only the exact traffic generated at the experiment time can be used, not allowing any traffic variation characteristics.

In contrast, *Fontes et al.* [13, 14] proposes a different method, replaying the signal physical characteristics, effectively replacing the physical layer and allowing the network simulator to handle all layers above it. Instead of using theoretical models of path loss to calculate the RSS, it uses values recorded during live experiments of RSS indicator (RSSI), an indication of the RSS observed by the network card. It benefits from having greater flexibility regarding the traffic generated by the nodes, as they do not significantly influence the signal's physical characteristics but are influenced by it. This method is not free from limitations, as it does not support any variation to the number, position or trajectory of the network nodes.

To tackle these limitations, a ML-based path loss model, as explained in Section 2.3.4, can be integrated with ns-3, allowing researchers to benefit from trace-based simulation advantages, while maintaining a degree of flexibility. The approach presented by *Zhang et al.* [53] was only evaluated in a urban environment, tests in other scenarios should be made to evaluate its performance in different environments. Additionally, it is too simplistic as the only features taken into consideration are the antenna-separation distance and the signal frequency. While adding more features does not guarantee a performance improvement, it would be interesting to explore the effect of using additional features. *Popescu et al.* [39, 38] explored two ANN models that can be interesting, however more tests should be done to improve ML-based models performance. The model proposed by *Cabral et al.* [8] is based on a network with a fixed base station and a moving UAV, as such the used features are aligned with that domain and can not be used in different situations. As the dataset considered was collected during a single experiment, the conclusions found may have some difficulties generalising to other environments. Additionally, the only hybrid approach considered was using the Friis Model, no other classical models were explored by the authors. Neither approach has been integrated and tested in a network simulator nor do they take into consideration the fast fading component of the path loss, only predicting the average value of RSSI measured both at the UAV and the Base Station (BS).

2.5 Conclusion

Machine learning is an evergrowing field of research, with strong statistical and mathematical basis, capable of inferring relationships in data that would be seriously difficult or impossible for humans to do. Machine learning can be applied to multiple domains, such as path loss estimation and classical wireless network problems.

Several approaches to estimate *RSS* in wireless network nodes have been proposed in the past, from simpler deterministic/theoretical models to more complex ones using machine learning. ML-based models had a better performance than the ones using traditional techniques [8, 53, 39, 38], which indicates a potential solution for improving network simulation while maintaining a degree of flexibility. However, the discussed approaches are only tested on specific scenarios and have not been integrated in network simulators.

Reinforcement Learning is a methodology growing in popularity, capable of dealing with several types of problems, ranging from playing games better than humans [31, 43] to improve the energy efficiency [34] or image transmission quality [21] of flying networks. Training an RL agent requires a dynamic environment, where the agent's actions influence and modify the environment, allowing it to explore multiple actions in different scenarios. As such, the improved accuracy of a trace-based simulation, with machine learning augmented data, is a potentially attractive solution to that problem. The state-of-the-art ML-based path loss models are a good starting point to dynamically generate the above mentioned scenarios. However, as previously discussed, those models share some drawbacks. Therefore, an improved solution that addresses those problems is fundamental.

Chapter 3

Problem and proposed solution

As previously stated in Section 1.3, this work's main goal is to create a trace-based ns3-gym framework, which supports developing, training and evaluating RL algorithms for wireless networks. To achieve the main goal, an intermediate step is to develop and integrate an ML trace-based model in the ns-3 simulator.

This chapter discusses an overview of the main problem and how the proposed solution will be accomplished.

3.1 Problem Definition

As stated in Sections 1.1 and 1.2, pure simulation – using classical propagation loss theoretical models – achieves results that can differ from those obtained in experimentation. Nevertheless, it is not feasible to only do experiments, as multiple conditions – such as, their setup cost and testbed availability –, make them impractical. Consequently, the need for a hybrid approach is evident, where the advantages of both techniques are merged, mitigating their disadvantages. All the past approaches to solve this problem fall short of this goal in some aspect, such as lack of flexibility, only replaying the exact conditions registered in the experiment [14] or the absence of integration with existing network simulators [8].

State-of-the-art trace-based simulations do not support any change to the environment, which is crucial in an RL agent's training. As such they can not be used in a trace-based ns3-gym framework. Pure simulation approaches are also not feasible, as the lack of accuracy would compromise the agent performance in real world applications. To develop such framework, a novel ML trace-based simulation able to dynamically generate scenarios similar to the one characterised by the collected traces has to be implemented.

3.2 Solution Architecture

In the RL agent training process, the agent has to observe the environment and act on it. As the environment is an ns-3 simulation, ns3-gym is used, acting as a bridge between ns-3 simulations and the RL agent. ns3-gym exposes an OpenAI Gym [6] interface through which the agent is able to communicate with the environment implementing the collection of the observations, the calculation of the reward and the execution of the action in the ns-3 simulation.

Figure 3.1 depicts the architecture of the proposed solution, with ns3-gym acting as a bridge between the ns-3 trace-based simulation and the RL agent.

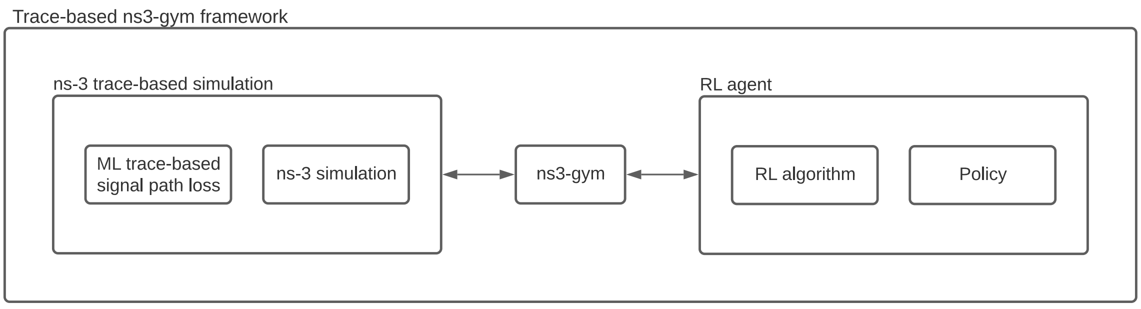

Trace-based ns3-gym framework
ns-3 trace-based simulation
ML trace-based signal path loss
ns-3 simulation
ns3-gym
RL agent
RL algorithm
Policy

Figure 3.1: Proposed trace-based ns3-gym framework architecture.

Figure 3.2 represents the steps to follow in order to build the proposed trace-based ns3-gym framework, organised in three major phases.

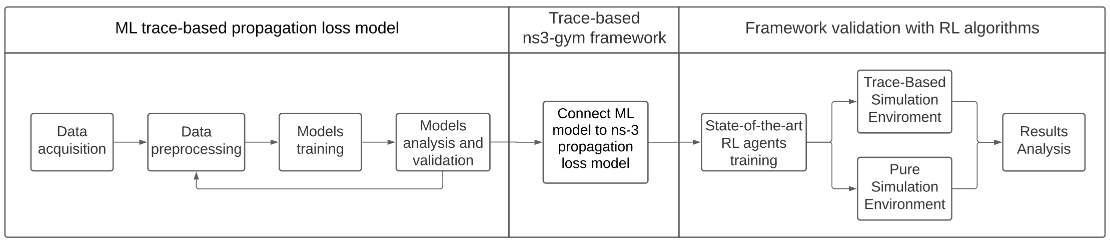

ML trace-based propagation loss model
Trace-based ns3-gym framework
Framework validation with RL algorithms
Data acquisition
Data preprocessing
Models training
Models analysis and validation
Connect ML model to ns-3 propagation loss model
State-of-the-art RL agents training
Trace-Based Simulation Enviroment
Pure Simulation Environment
Results Analysis

Figure 3.2: Sequence of steps for the development of the proposed solution.

The first step towards this goal is to design an ML trace-based model able to dynamically generate scenarios based on previously collected traces. Using the dataset of network traces collected in SIMBED [27] research project developed at INESC TEC, multiple machine learning algorithms were analysed and compared, looking for the model that most accurately replicates the Propagation loss values throughout the experiment.

The next step consists in integrating the propagation loss model in the ns-3 network simulator, as a custom *PropagationLossModel* able to communicate with a Python process via *ns3-ai* [51]. This setup has the advantage of having two different processes running ns-3 and the ML model, communicating via shared memory. This allows the use of standard ML frameworks, such as Tensorflow [12] or PyTorch [36].

After integrating the ML model in ns-3, it is possible to design network simulations in the same conditions as the experiments, even if the network topology differs from the recorded experiment. This improved accuracy helps an RL agent's training process, as it will benefit from more realistic results than a pure simulation approach. Finally, this idea shall be put in practice, validating it using state-of-the-art RL algorithms.

3.3 ML Trace-based Propagation Loss Model

The ML Trace-based Propagation Loss model simulation approach consists of two components, a path loss component and a fast-fading one. A ML model is combined with a fitted statistical distribution to predict the complete propagation loss value. The path loss component is responsible for calculating the main signal power loss that happens naturally as the distance increases, as the fast-fading component handles the signal reflections and other phenomena that alters the received power, even when the distance is constant. When a new propagation loss value is requested to the ML trace-based propagation loss model, given the distance between the nodes, the path loss component is predicted by the ML model and a pseudo-random value from the fast-fading distribution are added, getting the total propagation loss value, as depicted in Figure 3.3. This result is then combined with the transmission power to get the actual received power.

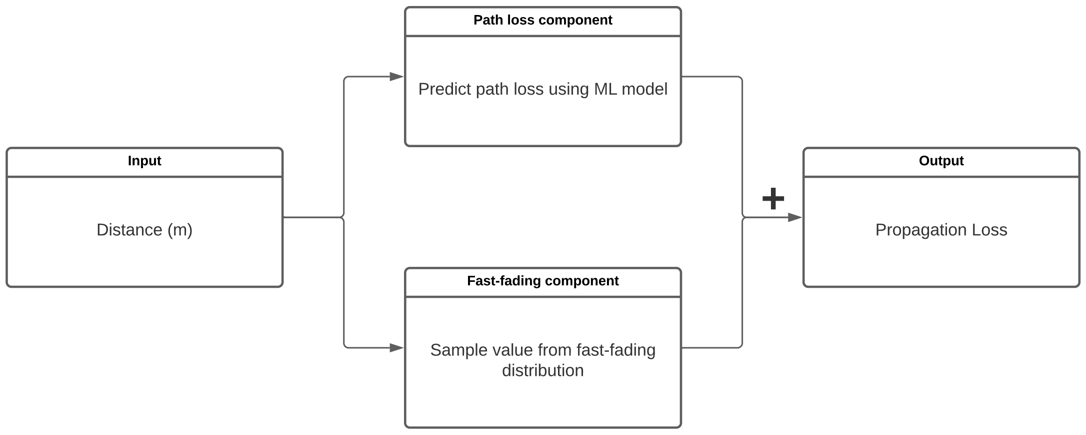


Figure 3.3: High-level diagram of the ML trace-based propagated loss model. The value of each component is calculated independently and then is summed, to achieve the complete propagation loss value.

3.3.1 Path Loss Model

To make the model independent from changes in transmission power, the collected SNR value was not used directly as training data. Instead, a preprocess step was done, where the path loss was calculated using Equation (3.1), considering an ambient noise N of -95 dBm, the receiver and transmitter antenna gains, G_r and G_t, and the transmission power P_t.

$$PL = -SNR - N + G_r + G_t + P_t \tag{3.1}$$

In order to train the ML model only on the Path Loss (PL) component, it is fundamental to isolate this component from the fast-fading. This was done by calculating the average propagation loss value per second. This inherently leads to a conjecture that the fast-fading component can be modelled as a Gaussian distribution with a mean of zero, as we are assuming that the average of the complete propagation loss values will counterbalance the fast-fading aspect. While there are other distributions more suitable for describing the fast-fading behaviour, such as Rician's and Rayleigh's [44], the process of isolating them from the PL component would be more challenging. Both *XGBoost* and *SVR* supervised learning models were trained on this data, with the objective of predicting the average PL value from the node's distance.

3.3.2 Fast-fading Model

To create the fast-fading model, each trace's propagation loss value was subtracted by the average PL value per meter. Normal, Rician and Rayleigh distributions were fitted to the data, the probability density function calculated and the sum of the squared error measured. The distribution with the smallest sum of squared error was selected as the fast-fading model to be used. The distribution fitting was done using the *scipy* distribution *fit* function, which finds estimates for distribution parameters that best adapt to the provided data. The same parameters were used independently of the distance, that is the fast-fading distribution remains uniform for all distances.

3.4 Trace-based ns3-gym framework

To take advantage of the previously engineered ML trace-based propagation loss model in ns-3, a connection between them must be established. The chosen approach relies on a custom *PropagationLossModel* that communicates with a Python process through shared memory, using *ns3-ai*. The development was done on ns-3 version 3.33 and all source code is publicly available [1].

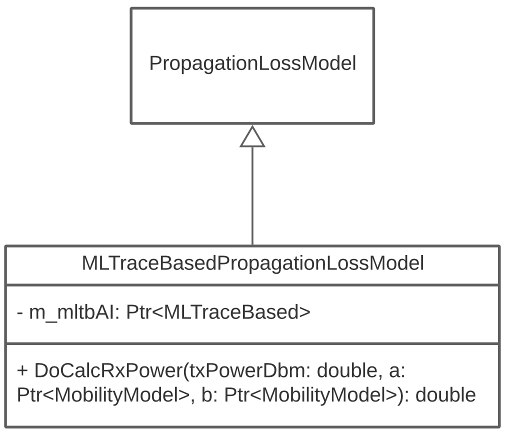

PropagationLossModel
MLTraceBasedPropagationLossModel
- m_mltbAI: Ptr<MLTraceBased>
+ DoCalcRxPower(txPowerDbm: double, a: Ptr<MobilityModel>, b: Ptr<MobilityModel>): double

Figure 3.4: Class diagram for the proposed *MLTraceBasedPropagationLossModel*.

The novel *MLTraceBasedPropagationLossModel*, characterised by the class diagram on Figure 3.4, is a subclass of ns-3's *PropagationLossModel*. The base class was extended, adding a pointer to a *MLTraceBased* instance and overriding the *DoCalcRxPower* method. *MLTraceBased*

[1] https://github.com/gregueiras/feup_tese

is a subclass of *Ns3AIRL*, an *ns3-ai* class that handles the communication between ns-3 and the Python process. This communication was defined by two C++ structures: *TbFeature* and *TbPredicted*, as can be seen in Figure 3.5.

```
struct TbFeature
{
  double distance;
}Packed;
```

```
struct TbPredicted
{
  double path_loss;
}Packed;
```

Figure 3.5: Structures used by ns3-ai to communicate to Python process.

TbFeature describes the features to be passed to the ML model, in this case, the only feature is the distance between the two nodes. *TbPredicted* represents the predicted propagation loss value.

In practice, when a new packet arrives to the node, its received power *RxPower* must be calculated. If it is using the *MLTraceBasedPropagationLossModel*, instead of directly calculating it, the features are passed to a Python process that uses the ML model to predict the propagation loss for the input distance and send it to the simulation, where it is added to the transmission power, to calculate the effective *RxPower*. This Python process starts by loading the ML models and then runs in a loop, waiting for new features to arrive. When that happens, it calculates the propagation loss and sends it back. This mechanism repeats until the simulation reaches its end.

Chapter 4

Trace-based ns3-gym framework validation

This chapter describes the experimental setup used to analyse and evaluate the solution discussed in the previous chapter.

4.1 Dataset

The used data was compiled in the context of the SIMBED project [27] where it was collected using Fed4FIRE+ [1] testbeds, in a warehouse environment. For SIMBED, four experiments were done, three of them only using static nodes and one using a static and a moving node. This was SIMBED's 3rd experiment, that was used to train and evaluate the ML trace-based propagation loss models, as it provided continuous distance values, as the dynamic node moved throughout the environment, while the other experiments were limited to the discrete values of the static nodes' distances. The collected traces registered multiple metrics, such as distance, SNR and goodput. Each experiment is comprised of multiple runs, varying the transmitter node transmission power from 0 dBm to 12 dBm.

4.2 ML Trace-based Propagation Loss Model Validation

In preparation for a validation of the ML trace-based propagation loss model in a ns-3 environment, some simulations were made. The ML models trained either in complete or partial data from an experiment run were evaluated and measured against pure simulation approaches in other runs. Those measurements were made by comparing the propagation loss predicted by the ML models with both the real measurements and the pure simulation calculations.

To test the ML models in different contexts, three scenarios were idealised. All three scenarios share a ground rule: the training data was collected in one experiment run and the test data came from a different run. In the "Full Set" scenario, the model was trained with all available data from a

[1] https://www.fed4fire.eu/

run and tested on another run. This should tell us if the model is able to extract knowledge from the data and learn from it. After that, the model was trained on shorter distances and tested on longer distances, to evaluate its extrapolation capabilities. Finally, the distances were split into some bins and the model learnt from some of them, effectively training on data with knowledge gaps. This scenario showcases the model interpolation behaviour. The training data [27] originated from run *08022019_11.04.35*, which is characterised by a transmission power of 1 dBm. The test data comes from run *07022019_02.49.27*, where a transmission power of 7 dBm was used. Both runs share the same channel centre frequency, 5220 MHz, bandwidth, 20 MHz, and a gain of -7 dBi was used for each antenna. The gain of the antennas is being represented as a negative value since signal attenuators of 10 dB were used in-line with the 3 dBi antennas, to limit the signal's range in the warehouse.

4.2.1 Full Set scenario

In the first scenario, where all data is available to training, both Support Vector Regression (SVR) and XGBoost have shown to better modulate the propagation loss when compared to either Friis or Log-distance path loss models, as can be seen in Figure 4.1.

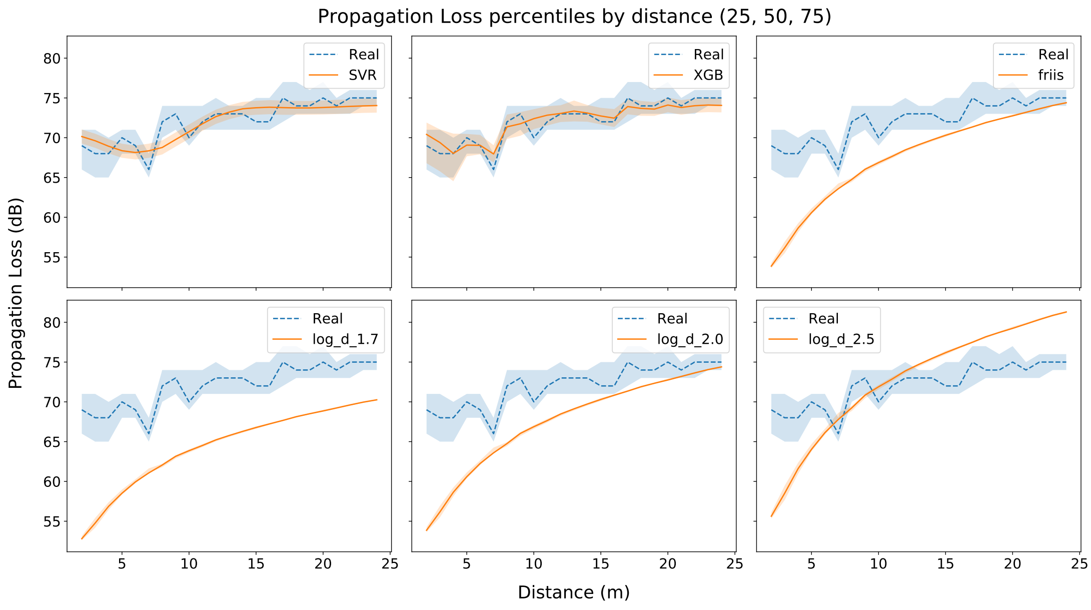


Figure 4.1: Propagation loss prediction by distance and propagation loss model. The thicker line represents the median, while the shadow bounds the 25th and 75th percentiles.

The thicker lines represent the median, while the shadow bounds the 25th and 75th percentiles. When a fast-fading component is not present, such as in Friis and Log-distance models, the 25th and 75th percentiles are not shown on the plot since these models are deterministic. XGBoost follows the real data closely, being able to replicate local spikes along the experiment, while SVR's curve is smoother. In this context, Friis and Log-distance with path loss exponent less than or equal to 2.0 are too optimistic while a path loss exponent of 2.5 is too pessimistic.

Figure 4.2 shows us the absolute difference between the 25th, 50th (median) and 75th percentiles for each model and the real data. The black dashed line serves as a baseline for the perfect scenario, where both models' percentiles have the same value. These follow the Figure 4.1 findings, where SVR and XGBoost have a smaller error. XGBoost difference is slightly smaller than the SVR one. Pure simulation models show a maximum error around 15 dB, while ML models errors' stay below 5 dB.

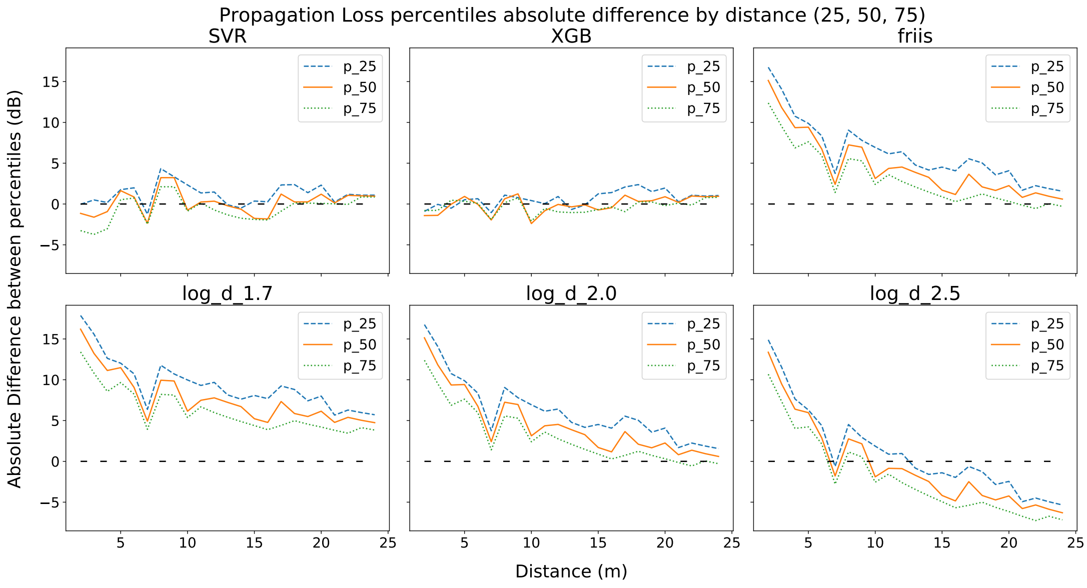


Figure 4.2: Propagation loss absolute difference by distance and model. Each line represents the absolute difference between the respective percentiles of the real data and the model in study. The black horizontal line represents the perfect scenario, where both the model and the real percentiles are the same.

The boxplot in Figure 4.3 represents the propagation loss distribution for each model, highlighting the median, the 25th and the 75th percentiles. The whiskers depict the full distribution, excluding some points considered to be outsiders, shown individually. The points are considered outliers if they fall outside of the area defined by multiplying the inter-quartile range, $Percentile_{75} - Percentile_{25}$, by 1.5, subtracting it to the 25th percentile and adding it to the 75th percentile. When comparing the propagation loss distribution for each model, in Figure 4.3, XGBoost and SVR present similar results, both being able to accurately model the real data. The analysis of the boxplot, in conjunction with the rest of the data, shows that the ML trace-based propagation loss model can correctly learn and represent the real experiment data when trained on the full dataset.

To choose the distribution that more accurately represents the fast-fading component, normal, Rician and Rayleigh distributions parameters were fitted to it, the probability density function calculated and the sum of the squared errors (SSE) compared, choosing the distribution with smallest value. Figure 4.4 represents the distributions of the multiple possible fast-fading distributions

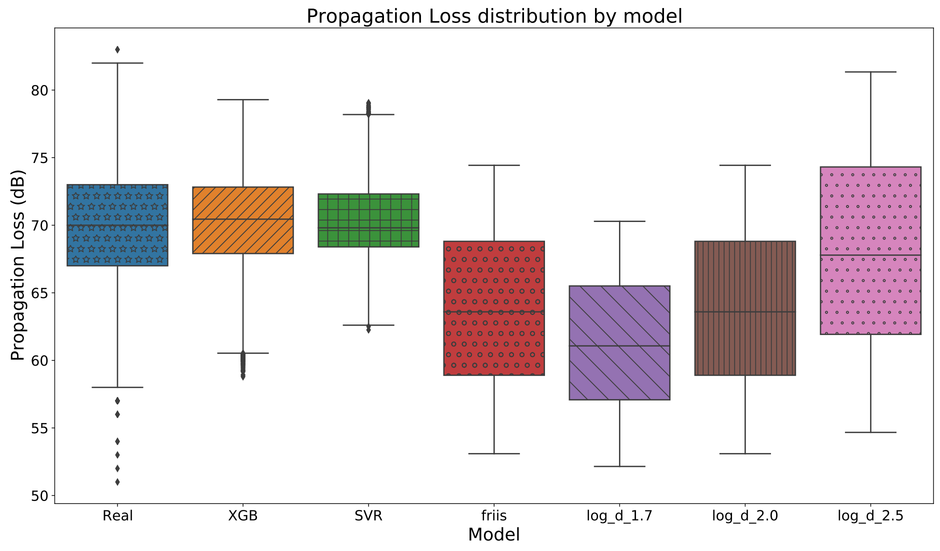


Figure 4.3: Propagation loss boxplot by model, showcasing the median, 25th percentile and the 75th percentile of the different distributions.

compared with the real one. The distribution that more accurately represents it is the normal distribution, as they are almost indistinguishable. The SSE values are 0.5, 7.3 and 8.0, for normal, Rician and Rayleigh's distributions, respectively.

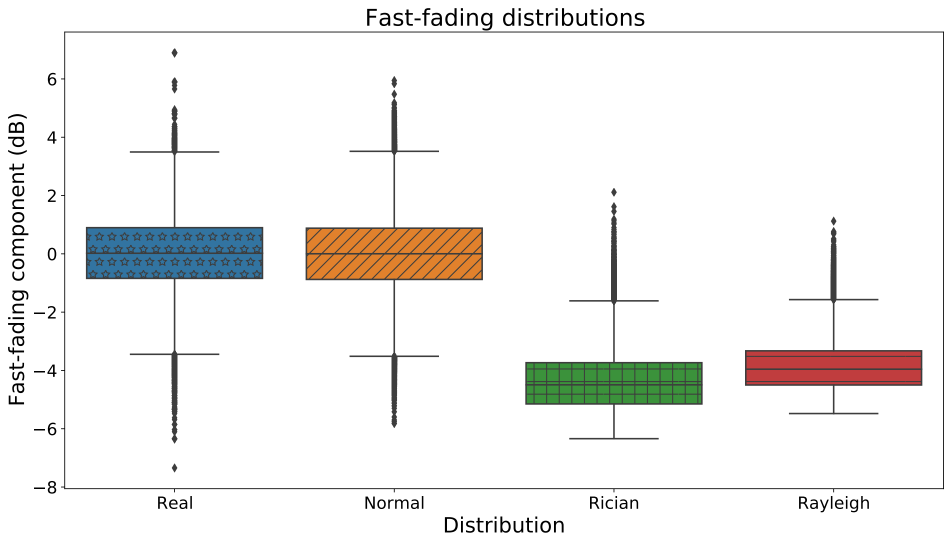


Figure 4.4: Fast-fading component boxplot by distribution, highlighting the median, 25th percentile and the 75th percentile for each distributions.

4.2.2 Extrapolation scenario

In the second scenario, where only the data until 10 m is available to training, the ML trace-based models' behaviour changes when compared to the "Full Set" scenario, as can be seen in Figure 4.5.

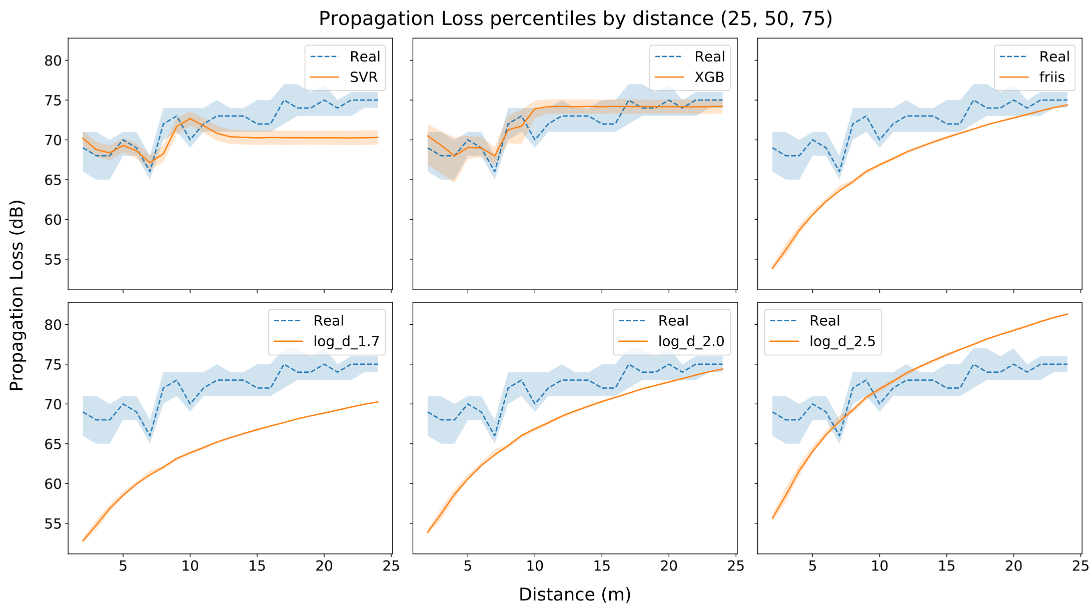


Figure 4.5: Propagation loss prediction by distance and propagation loss model. The thicker line represents the median, while the shadow bounds the 25th and 75th percentiles.

While predicting propagation loss for known distance values, the behaviour is similar to the previous experience, however, when dealing with unseen data, data not included in the training set, both ML trace-based models predict virtually a constant value. As expected, the pure simulation models' behaviour does not change.

Comparing the propagation loss distribution median, 25th and 75th percentiles absolute difference for each model in Figure 4.6, shows a similar situation, where the difference curve for unseen values follows the real data one, as the predicted values are constant.

When comparing the propagation loss values distribution, the results are similar to the previous scenario, with XGBoost providing the most accurate representation of the data.

Considering all information, it is possible to infer that the extrapolation capabilities of the ML trace-based propagation loss model are limited and other training strategies should be preferentially used.

Following the same methodology of the previous scenario, Rician and normal distributions have the same SSE, that is, they both represent the fast-fading component with the same accuracy, as can be observed in Figure 4.8. The distributions' SSE values are 0.8, 0.8 and 2.3 for normal, Rician and Rayleigh's distributions, respectively.

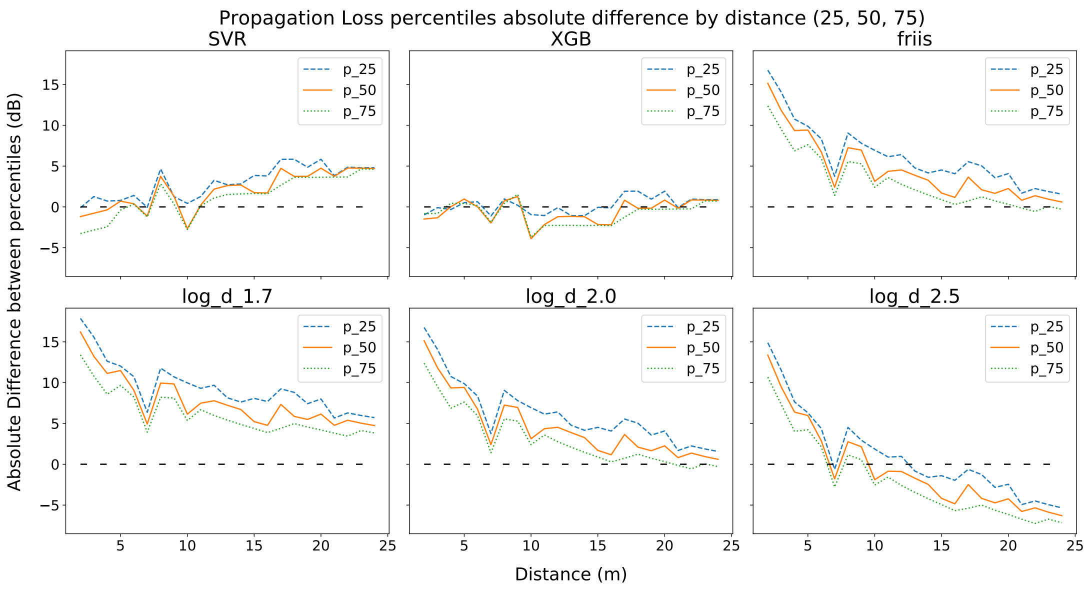


Figure 4.6: Propagation loss absolute difference by distance and model. Each line represents the absolute difference between the respective percentiles of the real data and the model in study. The black horizontal line represents the perfect scenario, where both the model and the real percentiles are the same.

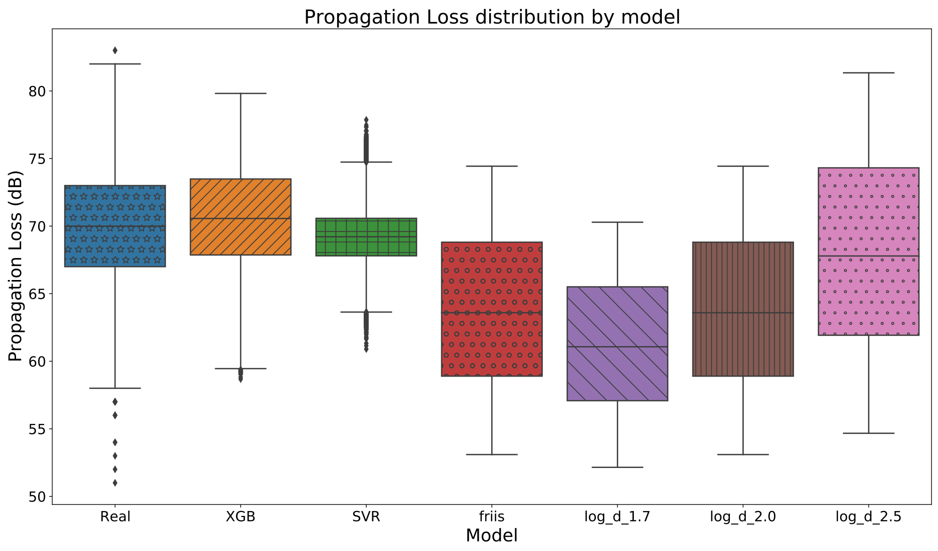


Figure 4.7: Propagation loss boxplot by model, showcasing the median, 25th percentile and the 75th percentile of the different distributions.

4.2.3 Interpolation scenario

In the last scenario, a different strategy was employed. The training data had some gaps, i.e. the training data consisted of data from distances shorter than 5 m, between 10 m and 15 m and larger

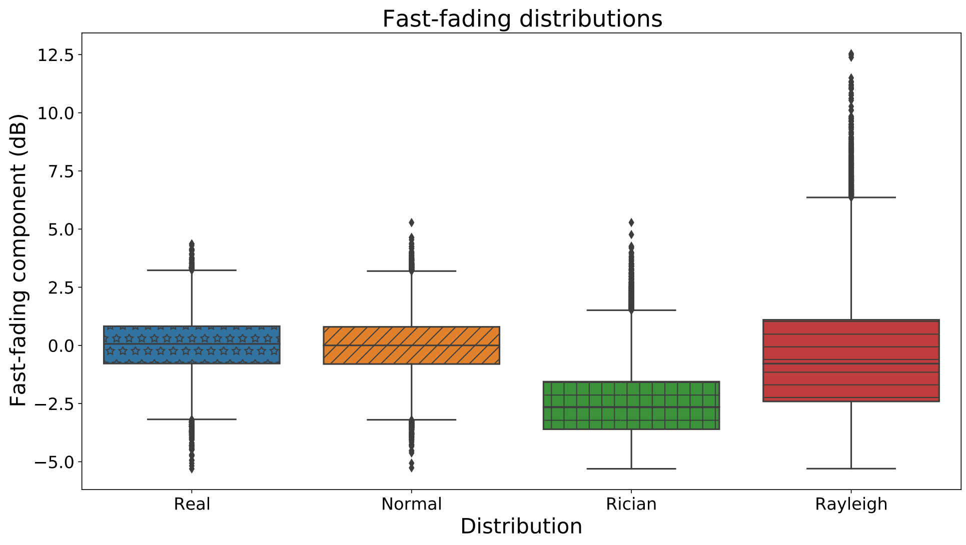


Figure 4.8: Fast-fading component boxplot by distribution, highlighting the median, 25th percentile and the 75th percentile for each distributions.

than 20 m. As can be seen in Figure 4.9, both SVR and XGBoost curves are essentially contained in the real data shadow, except on the local spikes excluded from train data, where both models follow the overall tendency on that spot. This indicates that this approach still allows us to achieve good performance while using less data.

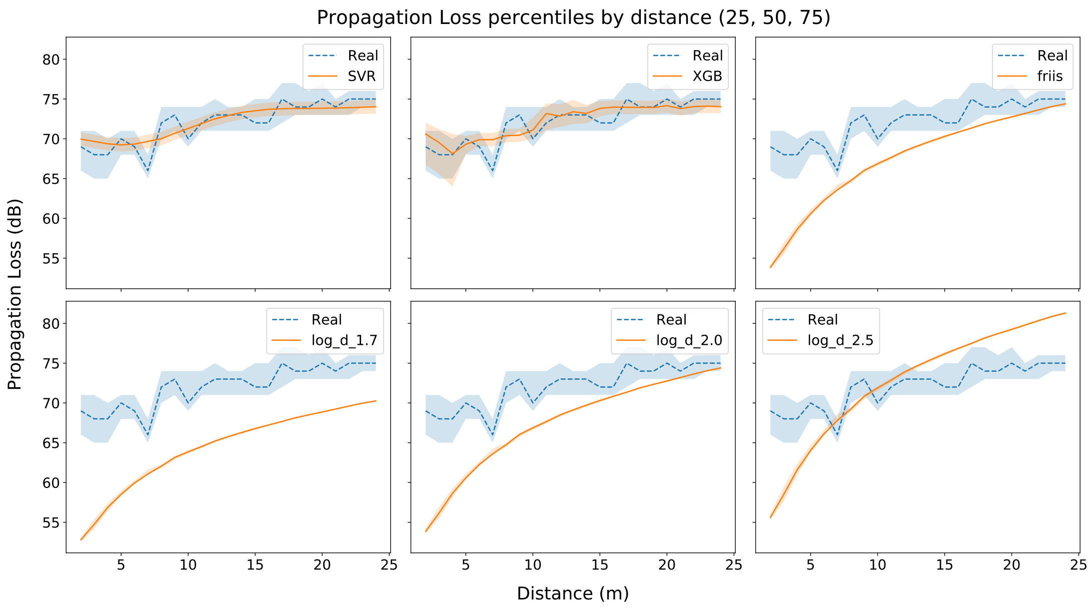


Figure 4.9: Propagation loss prediction by distance and propagation loss model. The thicker line represents the median, while the shadow bounds the 25th and 75th percentiles.

Looking at the propagation loss distribution median, 25th and 75th percentiles absolute difference for each model in Figure 4.10, it is possible to infer that the knowledge gaps were not a big problem for the ML models, as they were able to infer those missing values based on the previous and following ones.

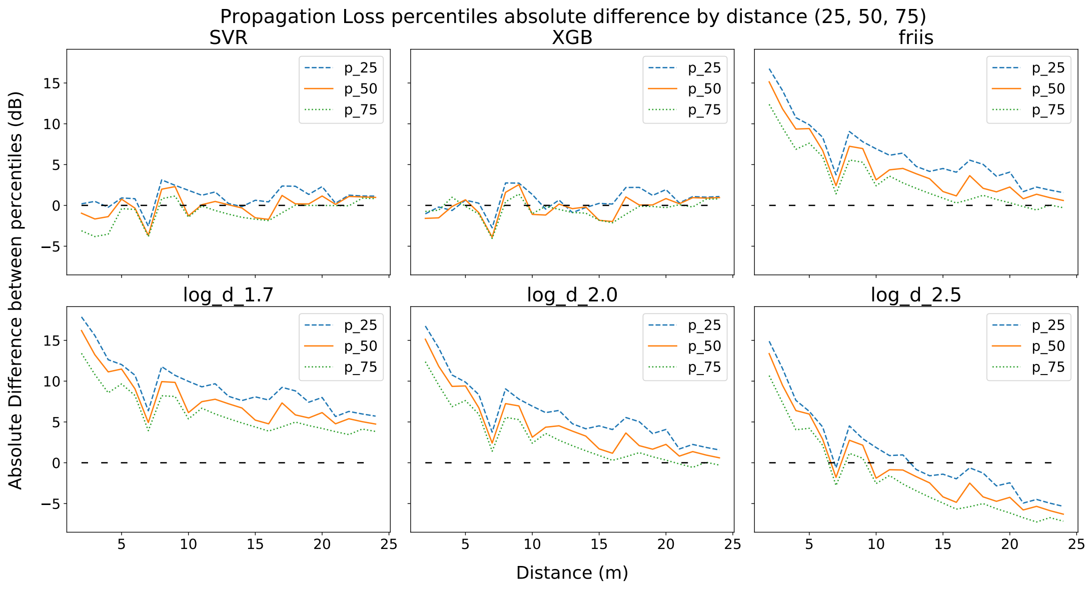


Figure 4.10: Propagation loss absolute difference by distance and model. Each line represents the absolute difference between the respective percentiles of the real data and the model in study. The black horizontal line represents the perfect scenario, where both the model and the real percentiles are the same.

The propagation loss distribution for each model is similar to the extrapolation scenario, where XGBoost and SVR distributions are more identical to the real one than the pure simulation models distributions, as can be observed in Figure 4.11.

Applying the same strategy for choosing the most accurate fast-fading component distribution as in the other two scenarios, the distribution chosen is the normal one, as can be observed in Figure 4.12. The calculated SSE values are 0.8, 2.6 and 2.4, for normal, Rician and Rayleigh's distributions, respectively.

4.3 ML Trace-based Propagation Loss Model impact in ns-3 simulation

To evaluate the impact of using the proposed *MLTraceBasedPropagationLossModel* on a specific wireless network scenario, several simulations on the scenario used to train the ML models were made. The wireless network performance on that scenario using pure simulation models and the novel ML trace-based propagation loss model were measured and compared. The simulations used one static and one moving node. The path taken by the moving node follows the path of

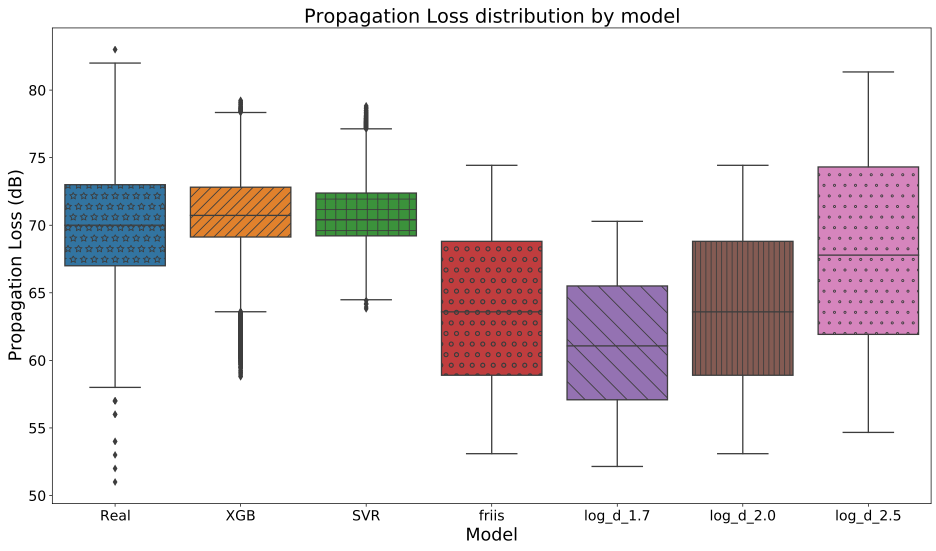


Figure 4.11: Propagation loss boxplot by model, showcasing the median, 25th percentile and the 75th percentile of the different distributions.

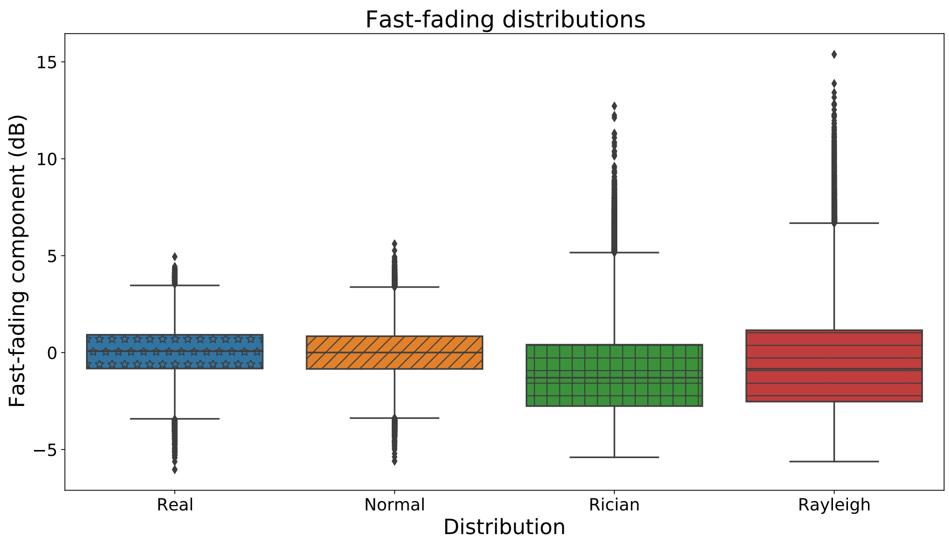


Figure 4.12: Fast-fading component boxplot by distribution, highlighting the median, 25th percentile and the 75th percentile for each distributions.

the test experiment run, *07022019_02.49.27*, and the simulation was configured to use the same parameters as that run. Those parameters can be consulted on Table 4.1. Both nodes used the *ConstantPositionMobilityModel* to ensure that their position was consistent with the original run. The moving node position was updated once a second, as that was the available distance data resolution. The fixed node generated a, UDP flow with a constant bitrate of 54 Mbit/s, to ensure that the connection was fully loaded, as the offered load was always above the link capacity, since we were using the IEEE 802.11a standard. *MinstrelWifiManager* was used to automatically adjust the data rate based on the link condition. The traffic was generated in simulation using the *OnOffApplication*, which was configured to be always on and transmitting packets of 1400 bytes. All unspecified parameters used the ns-3 default values.

Parameter	**Value**
ns-3 version	3.33
Wi-Fi standard	IEEE 802.11a
Preamble detection threshold (dBm)	-90.0
Packet Size (bytes)	1400
Generated bitrate traffic (MBit/s)	54
Duration (s)	404
Tx Power (dBm)	7
Frequency (MHz)	5220
Bandwidth (MHz)	20
Receiving Antenna Gain (dBi)	-7
Transmission Antenna Gain (dBi)	-7
Minimum Distance (m)	2.07
Maximum Distance (m)	24.09

Table 4.1: ns-3.33 simulation parameters. All unspecified parameters used the ns-3 default values.

To measure the Machine Learning Trace-based propagation loss model impact on that simulation scenario, it was compared against several pure simulation propagation loss models, such as Friis and Log-distance using different path loss exponents. The training approach followed the "Full Set" scenario, as it was the most accurate model. The Log-distance model was used in conjunction with *JakesPropagationLossModel*, in order to reproduce the fast-fading component of the propagation loss. As the Friis model and Log-distance with 2.0 exponent are identical, a fast-fading component was not added to the Friis model, leaving it fully deterministic; this way, the effect of the fast-fading component can be evaluated by comparing the Friis and Log-distance 2.0 (with fast fading) curves. The propagation loss models were compared using the goodput measured on the receiving node, that is the number of bits of useful information delivered to the application layer of the receiving node per second. A stronger received signal should reflect on a higher goodput, as Minstrel uses higher data rates to transmit the packets and as less errors occur and more messages can be successfully received in a given time frame and vice-versa, making this a suitable metric to evaluate the impact of the proposed ML trace-based propagation loss model. As the transport protocol used was UDP, erroneous messages or messages that could not be delivered are simply dropped and do not count towards the measured goodput.

The real experiment data can be seen in Figure 4.13. As expected, the goodput and the distance are inversely proportional. As the distance increases the signal gets weaker and the goodput decays. When the distance decreases, we can observe the opposite, a goodput increase.

In the real-world experiment, the mobile node starts approximately 5 m away from the static

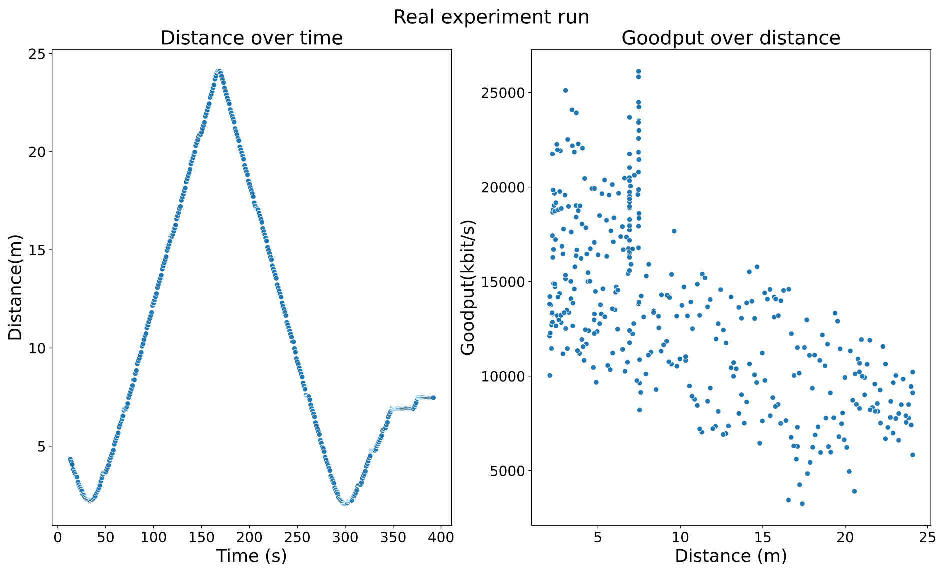


Figure 4.13: Real run goodput over distance.

one, moves in its direction and then moves farther away, to a maximum of around 25 m. After reaching this point, it goes back to the static node position and finally distantiates itself again, ending the experiment roughly 7 m apart. The goodput does not follow this variation directly, as the small changes in distance at lower values do not reflect on the goodput. Instead, the goodput follows the significant changes in distance, achieving its peak value when the distance is smaller and its lowest at the most distant point.

In order to calculate statistically confident results, five runs were done for each model and the median of the values per second was calculated. As some of the results tend to be more disperse, a rolling window of five seconds was used to present the data, where the median of those values was calculated and used to represent each window.

The median of the goodput over distance for all propagation loss models is shown in Figure 4.14. Observing the results, we can infer that neither Log-distance nor Friis models are able to accurately reproduce the observed goodput, as the points are condensed, lacking the spread of the real data. Friis ends up being too optimistic, saturating the channel when the nodes distance is smaller, resulting in higher than expected goodput values during all experiment. SVR, while not completely replicating the original results, generally follows the trend of the real data, in which the goodput is inversely proportional to the distance. XGBoost, while being able to replicate some of the observed real values, is not able to accurately reproduce the goodput when the distance increases. However, it is the only model that accurately reproduced the spread nature of the real data. Both ML trace-based propagation loss models do not achieve as low goodput values as the real experiment, however, for higher values they tend to be more accurate. This data indicates that both ML models lead to more accurate representations of the real-world scenario than the pure simulation models.

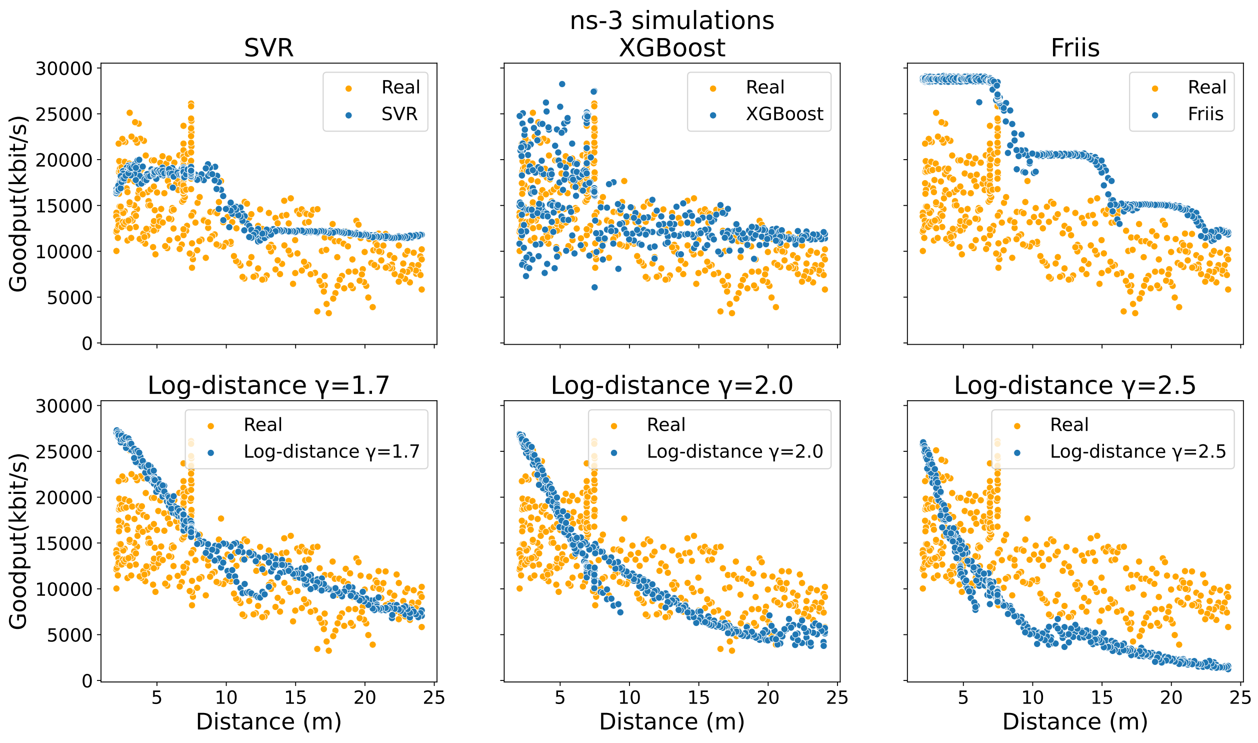


Figure 4.14: ns-3 simulations using different propagation loss models. All models predict both path loss and fast-fading components, except for Friis that only predicts the path loss component.

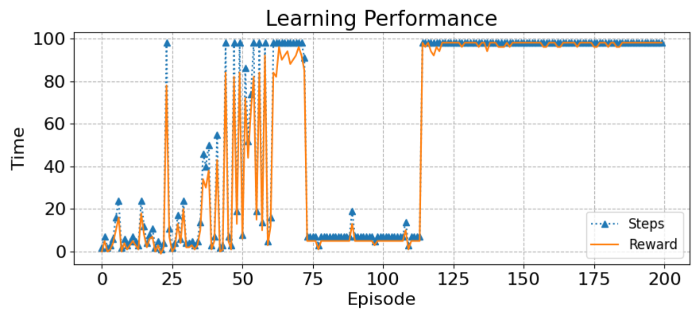
4.4 Trace-based ns3-gym framework validation

In order to validate the integration of the ML trace-based propagation loss model on the trace-based ns3-gym framework, an example from ns3-gym was used [18]. The problem proposed in this example concerns the selection of which radio channel to broadcast, between two nodes, in a environment where the interference follows a periodic pattern, switching through all channels. The nodes are distanced from each other by 10 m. An RL agent is trained with the objective of predicting which channel to use in the next time slot avoiding any collisions with the interference signal. This scenario was chosen by several reasons, as it is a state-of-the-art example, directly supported by ns3-gym and it depicts a wireless network scenario where the algorithm's performance directly depends on a good representation of the environment, which includes the propagation loss.

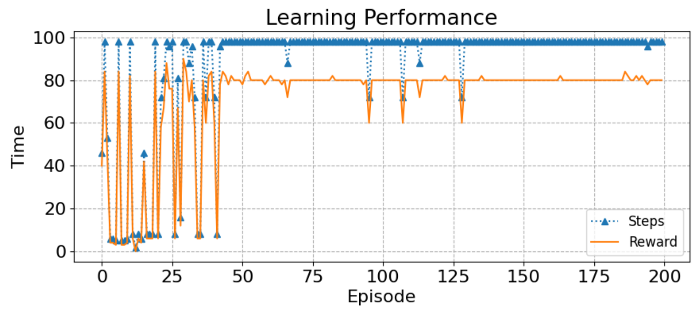
The simulation scenario provided by ns3-gym used the Friis propagation loss model, without fast-fading, and its performance was compared to both proposed trace-based ns3-gym frameworks using the SVR and XGBoost propagation loss models.

Each agent was trained for two hundred episodes, where in each episode it had to choose in which radio channel to transmit in the next time slot. For each correct decision it received a reward of 1 point, whereas if it chose an occupied channel the reward value was -1. If the agent accumulated 3 errors in the last 10 time slots, the episode would end. Each episode had at most 100 time slots. Figure 4.15 represents the learning performance of three agents, trained on the same scenario using different propagation loss models. As a baseline, the performance using the Friis model, without fast-fading, is shown on Figure 4.15a. The learning performance in the SVR and XGBoost trace-based propagation loss models scenarios is represented by Figure 4.15b and Figure 4.15c, respectively. The reward value obtained in each episode by the agent can be observed in the orange line, while the blue line with triangles indicate how much time slots the episode lasted. A perfect performance would be both lines at the top, as the agent was able to accurately learn the environment and make the correct decision every time. A plot with distance between both lines indicates that the agent made some mistakes, but not consecutively.

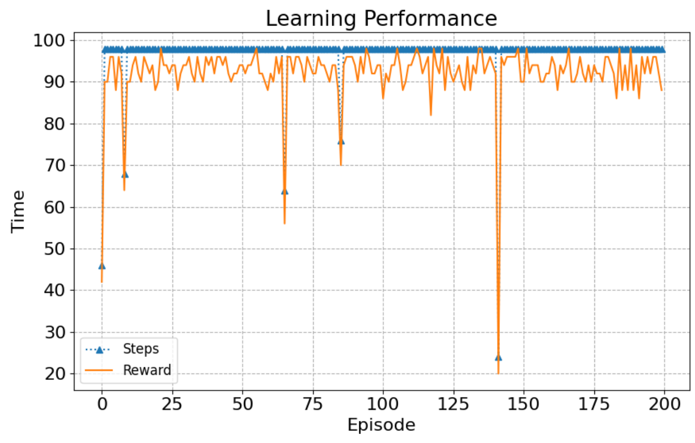
Comparing the data in Figure 4.15, we can observe that the RL agent learning from the simulation that used the Friis model was able to learn the environment and always select the unoccupied channel in around 120 episodes, while the agent that learnt from an environment backed by the ML trace-based propagation loss model was able to almost instantly learn it when using the XGBoost model. The simulation based on the SVR model, albeit converging faster than the Friis one, in around 45 episodes, was not able to reach the reward values as the other two approaches. This goes alongside with the expected results, as XGBoost was the most accurate model in the previous scenarios, Section 4.2 and Section 4.3. In principal, an agent trained on the trace-based ns3-gym framework would perform better in the real-world environment for which it was trained (based on real traces) than an agent trained on an environment based on pure simulation models.



(a) Learning performance RL agent trained on ns3-gym environment based on Friis propagation loss model.



(b) Learning performance RL agent trained on ns3-gym environment based on SVR trace-based propagation loss model.



(c) Learning performance RL agent trained on ns3-gym environment based on XGBoost trace-based propagation loss model.

Figure 4.15: RL agent training performance on the same scenario using different propagation loss models.

Chapter 5

Conclusions and Future Work

In this chapter we make an overview of the work developed, we reinstate the major original contributions, point out limitations of the developed framework and reflect on possible future work.

5.1 Overview of the Work Developed

The main goal of this dissertation was to improve trace-based network simulation, providing more flexibility and integrating it with the ns-3 simulator and ns3-gym, creating a trace-based ns3-gym framework that can be used to train and evaluate RL algorithms for wireless networks. In spite of the work being developed over a single dataset, it is made to be as flexible as possible and it could easily be used in other scenario/environment, provided that new traces are collected in that environment and the ML trace-based propagation loss models are trained on it.

This dissertation began by providing the context and motivation for why both simulations and experiments have limitations, leading to the necessity of a hybrid solution. These limitations go from the cost and complexity of setting up real-world testbeds to the lack of flexibility presented by state-of-the-art trace-based solutions. This constraint became even more relevant nowadays, that RL algorithms are being more applied to solve different wireless networks problems. Following that, a review of the related work and the state-of-the-art fields related to this dissertation was discussed. Machine Learning fundamental concepts were discussed, and some popular supervised and reinforcement learning concepts and algorithms were introduced. Propagation loss models were also presented, ranging from classical models to machine learning approaches. It was concluded that, to the best of our knowledge, no other proposed approach solves all the raised problems. The following chapter described the problem in detail, proposed a solution, and discussed how to implement it. Finally, the fourth chapter analysed the performance of the ML trace-based ns3-gym framework, validating all components of the proposed solution. First, the ML trace-based propagation loss models were compared against classical propagation loss models. After that, the models were integrated into ns-3 and their operation was evaluated, where ML trace-based propagation loss models showed to more accurately represent the real-world scenario than their pure simulation (either deterministic or stochastic) counterparts, even when both are used in conjunction. Finally, the complete trace-based ns3-gym framework was put to the test to solve a ”cognitive radio” problem using RL agents, which benefited from the improved accuracy of the proposed ML trace-based propagation loss models. The main objective was accomplished, as the trace-based ns3-gym framework was implemented and the ML trace-based propagation loss model is a better representation of the real scenario than the classical models, and its integration on ns-3 confirms that. The integration with ns3-gym also shows that the RL agents benefit from more accurate representations of the environment provided by the models mentioned above, which meant superior learning performance, as the agents were able to learn faster when compared to agents trained on environments using pure simulation propagation loss models. Finally, although not in the scope of this dissertation, such RL agents trained using the proposed trace-based ns3-gym framework should also perform better than the RL agents trained with pure simulation models, when deployed in the real-word scenarios for which they were trained/specialised for.

5.2 Original Contributions

The main original contributions of this work are the following:

1. **ML trace-based propagation loss model.** A ML model able to learn both the primary and the fast-fading component of the path loss from traces collected in an environment. This strategy allows the model to replicate the conditions of the environment, not only repeating the collected traces but rather be the basis for the creation of new ones in a virtual representation of the original environment.

2. **Trace-based ns3-gym framework.** An ns3-gym framework, where users can leverage the earlier mentioned models, combining the capabilities of the ns-3 simulator and the easy to use interface of ns3-gym to train and develop new RL algorithms and environments.

5.3 Future Work

In the following, we reflect on future work that would aggregate value to the trace-based ns3-gym framework and further explore its capabilities.

In the ML trace-based propagation loss model training process, instead of only using the distance between two nodes to predict the propagation loss value, it could be interesting to use other features, such as the node’s position in a Cartesian plane or its height. The fast fading component could also benefit from the position of the nodes, as it would enable us to adapt it to different parts of the environment, for example by dividing it in a grid, where each cell has its own fast fading component.

It is possible to use the implemented trace-based ns3-gym framework to create new models tailored for different environments. However, in order to improve the framework developed some refinements could still be considered, for example, how to overcome the difficulty in extrapolation, as noted in Section 4.2.2.

It would also be interesting to evaluate the performance of an agent trained with the proposed trace-based ns3-gym framework, in the same environment where its traces were collected, comparing it with the performance of a RL agent trained only on pure simulation environments.

Finally, the results achieved by the ML trace-based propagation loss model and by the trace-based ns3-gym framework could be published in a conference article. The framework could be made available in the ns-3 app store, associated with the ”Trace-based simulation” application group, currently in development by the wireless networks team in INESC TEC, in which this dissertation is included.

References

[1] Prakash Agrawal and Mythili Vutukuru. Trace based application layer modeling in ns-3. In *2016 Twenty Second National Conference on Communication (NCC)*, pages 1–6, Guwahati, India, March 2016. IEEE.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, March 1995.

[3] O. G. Aliu, A. Imran, M. A. Imran, and B. Evans. A Survey of Self Organisation in Future Cellular Networks. *IEEE Communications Surveys Tutorials*, 15(1):336–361, 2013. Conference Name: IEEE Communications Surveys Tutorials.

[4] A Baidya and Siladitya Sen. Qualitative Analysis on Log-Distance Propagation Model for Wlan Standard Engineering. *PARIPEX*, 1.6714, March 2014.

[5] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[7] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[8] João Rafael de Figueiredo Cabral. A Machine Learning Approach for Path Loss Estimation in Emerging Wireless Networks. February 2019. Accepted: 2020-02-03T03:54:33Z.

[9] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco California USA, August 2016. ACM.

[10] Christoph Dann, Gerhard Neumann, and Jan Peters. Policy Evaluation with Temporal Differences: A Survey and Comparison (Extended Abstract). *Proceedings of the International Conference on Automated Planning and Scheduling*, 25(1), April 2015. Number: 1.

[11] Department of Electronics & Communication Engineering Heritage Institute of Technology, Kolkata, India, A. Pallob Baidya, and B. Prof. Siladitya Sen. Qualitative Analysis on Log-Distance Propagation Model for Wlan Standard. *Paripex - Indian Journal Of Research*, 3(3):69–70, January 2012.

[12] TensorFlow Developers. Tensorflow, June 2021. Specific TensorFlow versions can be found in the "Versions" list on the right side of this page.<br>See the full list of authors <a href="https://github.com/tensorflow/tensorflow/graphs/contr ibutors">on GitHub</a>.

[13] Helder Fontes, Rui Campos, and Manuel Ricardo. A Trace-based ns-3 Simulation Approach for Perpetuating Real-World Experiments. In *Proceedings of the Workshop on ns-3 - 2017 WNS3*, pages 118–124, Porto, Portugal, 2017. ACM Press.

[14] Helder Fontes, Rui Campos, and Manuel Ricardo. Improving the ns-3 *TraceBasedPropagationLossModel* to support multiple access wireless scenarios. In *Proceedings of the 10th Workshop on ns-3*, WNS3 '18, pages 77–83, New York, NY, USA, June 2018. Association for Computing Machinery.

[15] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[16] H.T. Friis. A Note on a Simple Transmission Formula. *Proceedings of the IRE*, 34(5):254–256, May 1946.

[17] Piotr Gawłowicz and Anatolij Zubow. ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, November 2019.

[18] Piotr Gawłowicz and Anatolij Zubow. ns3-gym: Extending OpenAI Gym for Networking Research. *arXiv:1810.03943 [cs]*, October 2018. arXiv: 1810.03943.

[19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[20] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang. Deep-Reinforcement-Learning-Based Optimization for Cache-Enabled Opportunistic Interference Alignment Wireless Networks. *IEEE Transactions on Vehicular Technology*, 66(11):10433–10445, November 2017.

[21] Chen Hou, Zhexin Xu, Wen-Kang Jia, Jianyong Cai, and Hui Li. Improving aerial image transmission quality using trajectory-aided OLSR in flying ad hoc networks. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):140, December 2020.

[22] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[23] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010.

[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

[25] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani. The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective. *IEEE Wireless Communications*, 24(3):146–153, June 2017. Conference Name: IEEE Wireless Communications.

[26] Guolin Ke, Qi Meng, Thomas Finely, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIP 2017)*, December 2017.

[27] Vitor Lamela, Helder Fontes, Tiago Oliveira, José Ruela, Manuel Ricardo, and Rui Campos. SIMBED - Offline Real-World Wireless Networking Experimentation using ns-3, April 2019. Version Number: 1.0 type: dataset, https://zenodo.org/record/2634272.

[28] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang. TACT: A Transfer Actor-Critic Learning Framework for Energy Saving in Cellular Radio Access Networks. *IEEE Transactions on Wireless Communications*, 13(4):2000–2011, April 2014. Conference Name: IEEE Transactions on Wireless Communications.

[29] X. Liu, Y. Liu, and Y. Chen. Reinforcement Learning in Multiple-UAV Networks: Deployment and Movement Design. *IEEE Transactions on Vehicular Technology*, 68(8):8036–8049, August 2019.

[30] Q. Mao, F. Hu, and Q. Hao. Deep Learning for Intelligent Wireless Networks: A Comprehensive Survey. *IEEE Communications Surveys Tutorials*, 20(4):2595–2621, 2018.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari With Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*. 2013.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. Number: 7540 Publisher: Nature Publishing Group.

[33] João Mendes Moreira, André C. P. L. F. de Carvalho, and Tomáš Horváth. *A General Introduction to Data Analytics*. John Wiley & Sons, Inc., Hoboken, NJ, USA, June 2018.

[34] Yiwen Nie, Junhui Zhao, Jun Liu, Jing Jiang, and Ruijin Ding. Energy-efficient UAV trajectory design for backscatter communication: A deep reinforcement learning approach. *China Communications*, 17(10):129–141, October 2020.

[35] P. Owezarski and N. Larrieu. A trace based method for realistic simulation. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 4, pages 2236–2239 Vol.4, June 2004.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[37] Caleb Phillips, Douglas Sicker, and Dirk Grunwald. A Survey of Wireless Path Loss Prediction and Coverage Mapping Methods. *IEEE Communications Surveys & Tutorials*, 15(1):255–270, 2013.

[38] I. Popescu, D. Nikitopoulos, P. Constantinou, and I. Nafornita. Comparison of ANN Based Models for Path Loss Prediction in Indoor Environment. In *IEEE Vehicular Technology Conference*, pages 1–5, September 2006. ISSN: 1090-3038.

[39] Ileana Popescu, Dimitris Nikitopoulos, Philip Constantinou, and Ioan Nafornita. ANN Prediction Models for Outdoor Environment. In *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, Helsinki, September 2006. IEEE.

[40] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Englewood Cliffs, N.J, 1995.

[41] Takaya Saito and Marc Rehmsmeier. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10(3):e0118432, March 2015.

[42] Shi Zhong, T. M. Khoshgoftaar, and S. V. Nath. A clustering approach to wireless network intrusion detection. In *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 7 pp.–196, November 2005. ISSN: 2375-0197.

[43] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[44] A. Sudhir Babu and K.V Sambasiva Rao. "Evaluation of BER for AWGN, Rayleigh and Rician Fading Channels under Various Modulation Schemes". *International Journal of Computer Applications*, 26(9):23–28, July 2011.

[45] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao. Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues. *IEEE Communications Surveys Tutorials*, 21(4):3072–3108, 2019. Conference Name: IEEE Communications Surveys Tutorials.

[46] S. Tamoor-ul Hassan, S. Samarakoon, M. Bennis, M. Latva-aho, and C. S. Hong. Learning-Based Caching in Cloud-Aided Wireless Networks. *IEEE Communications Letters*, 22(1):137–140, January 2018. Conference Name: IEEE Communications Letters.

[47] Susan R. Wallace and F. Layne Wallace. Two neural network programming assignments using arrays. *ACM SIGCSE Bulletin*, 23(1):43–47, March 1991.

[48] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo. Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks. *IEEE Communications Surveys Tutorials*, 22(3):1472–1514, 2020.

[49] J. Wang, J. Wang, Y. Wu, J. Wang, H. Zhu, M. Lin, and J. Wang. A Machine Learning Framework for Resource Allocation Assisted by Cloud Computing. *IEEE Network*, 32(2):144–151, March 2018. Conference Name: IEEE Network.

[50] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari. Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):257–265, June 2018.

[51] Hao Yin, Pengyu Liu, Keshu Liu, Liu Cao, Lytianyang Zhang, Yayu Gao, and Xiaojun Hei. Ns3-ai: Fostering artificial intelligence algorithms for networking research. In *Proceedings of the 2020 Workshop on Ns-3*, WNS3 2020, page 57–64, New York, NY, USA, 2020. Association for Computing Machinery.

[52] L. Zhang, H. Zhao, S. Hou, Z. Zhao, H. Xu, X. Wu, Q. Wu, and R. Zhang. A Survey on 5G Millimeter Wave Communications for UAV-Assisted Wireless Networks. *IEEE Access*, 7:117460–117504, 2019.

[53] Yan Zhang, Jinxiao Wen, Guanshu Yang, Zunwen He, and Jing Wang. Path Loss Prediction Based on Machine Learning: Principle, Method, and Data Expansion. *Applied Sciences*, 9(9):1908, May 2019.