ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo 37

Settore Concorsuale: 09/F2 - TELECOMUNICAZIONI

Settore Scientifico Disciplinare: ING-INF/03 - TELECOMUNICAZIONI

DEEP LEARNING FOR INTELLIGENT AND AUTONOMOUS WIRELESS NETWORKS

Presentata da: Lorenzo Mario Amorosa

Coordinatore Dottorato

Davide Dardari

Supervisore

Roberto Verdone

Co-supervisore

Francesco Mete

Esame finale anno 2025

Abstract

In recent years, distributed and ubiquitous intelligence has become a central force driving groundbreaking advances in wireless networks. Enabled by emerging autonomy, next-generation wireless networks are set to undergo substantial evolution. The main objective for the future consists in designing and creating cohesive communication and learning frameworks to achieve intelligent and autonomous wireless networks, pursuing the ambitious goal of achieving human-out-of-the-loop artificial intelligence (AI). Meeting this challenge signifies a critical transformation toward AI-native wireless networks that can dynamically self-adapt to complex scenarios. Addressing these future challenges requires a comprehensive integration of extensive knowledge in fields such as wireless communication and deep learning, creating opportunities for data-driven wireless network design, management, and optimization.

This thesis investigates promising areas in wireless communications where AI paves the way for the achievement of intelligent and autonomous wireless networks. The first chapter introduces adaptive optimization in wireless communication networks and addresses the goal of learning effective and robust radio resource management strategies in complex scenarios. The second chapter explores the impact of generative AI in next-generation wireless networks, focusing on reliable and uncertainty-aware data generation processes enabled by approximate Bayesian learning. It illustrates how generative AI can represent an effective means of aiding data-driven algorithms in generalization and reducing the need for costly data collection. The third chapter then delves into distributed learning over wireless networks for radio resource management, which has the potential to meet the scalability demands of modern data-driven applications. This chapter emphasizes the importance of incorporating graph structures as an efficient way to introduce a relational inductive bias into the learning process, thus enhancing system performance across various metrics. Finally, the last chapter presents a holistic performance analysis of AI-native applications in 5G industrial internet-of-things networks, particularly within safety-critical scenarios, highlighting practical considerations on AI-native wireless network architectures.

Contents

List of Acronyms

1D-CNN 1D convolutional neural network

3GPP 3rd generation partnership project

5CN 5G core network

5G NR 5G New Radio

5G-ACIA 5G Alliance for Connected Industries and Automation

5G 5th generation

ACK acknowledgment

AE autoencoder

AGV automated guided vehicle

AI artificial intelligence

AL active learning

ANN artificial neural network

APN access point name

AP access point

AQoSA agile QoS adaptation

AoI age of information

B5G beyond 5G

BDL Bayesian deep learning

BER bit error rate

BLER block error rate

BL Bayesian learning

BNN Bayesian neural network

BN Bayesian network

BS base station

BiLSTM bi-directional long short term memory

CCO capacity and coverage optimization

CDR call data record

CE calibration error

CIR channel impulse response

CNN convolutional neural network

CN core network

CQI channel quality indicator

CRS channel reference signal

CSI channel state information

CTCE centralized training-centralized execution

CTDE centralized training-decentralized execution

CTF channel transfer function

DCI downlink control information

DDPG deep deterministic policy gradient

DL downlink

DNN deep neural network

DQN deep Q-network

DRL deep reinforcement learning

DTDE decentralized training-decentralized execution

E2E end-to-end

EDF earliest deadline first

ELBO evidence lower bound

ERM empirical risk minimization

FC-NN fully-connected neural network

FF fairness first

FIFO first in first out

FL federated learning

FR1 frequency range 1

FR2 frequency range 2

G-KDE Gaussian kernel density estimation

GAI generative artificial intelligence

GAN generative adversarial network

GBR guaranteed bit-rate

GCN graph convolutional network

GML machine learning on graphs

GNN graph neural network

GPS global positioning system

GRL graph reinforcement learning

GRU gated recurrent unit

HARQ hybrid automatic repeat request

HetNet heterogeneous network

IDS intrusion detection systems

IDs identifier

IIoT industrial internet-of-things

IPW inverse probability weighting

InF indoor factory

IoT internet-of-things

KDE kernel density estimation

KPIs key performance indicators

KPI key performance indicator

LAT latitude

LLM large language model

LNA low noise amplifier

LON longitude

LR logistic regression

LSTM long-short term memory

LTE long-term evolution

LoS line-of-sight

MAC medium access control

MAE mean absolute error

MAP maximum a posteriori

MARL multi-agent reinforcement learning

MCS modulation and coding scheme

MDP Markov decision process

MDT minimization of drive test

MEC multi-access edge computing

MIMO multiple-input multiple-output

MISE mean integrated squared error

MLB mobility load balancing

MLE maximum likelihood estimation

MLP multi layer perceptron

ML machine learning

MMSE minimum mean squared error

mmWave millimeter wave

MNO mobile network operator

MSE mean square error

NG-RAN Next Generation radio access network (RAN)

NLP natural language processing

NLoS non line-of-sight

NN neural network

NOMA non-orthogonal multiple access

NPN on-net non public network on-net

NPN on-premise non public network on-premise

NPN non public network

NR new radio

OFDMA orthogonal frequency division multiple access

OFDM orthogonal frequency division multiplexing

PA power amplifier

PBNN probabilistic bayesian neural network

PCELL primary cell

PCI physical cell ID

PCP Poisson cluster process

PDCCH physical downlink control channel

PDCP packet data convergence protocol

PDL probabilistic deep learning

PDSCH physical downlink shared channel

PDU protocol data unit

PHY physical

PLC programmable logic controller

PL path loss

PNN probabilistic neural network

PN public network

PO-MDP partially observable Markov decision process

PPO proximal policy optimization

PPP Poisson point process

PQoS predictive quality of service

PRBS pseudorandom binary sequence

PRB physical resource block

PS parameter server

PUCCH physical uplink control channel

PUSCH physical uplink shared channel

QCI QoS class identifier

QoS quality of service

RACH random access channel

RAN radio access network

RAN radio access network

RB resource block

RE resource element

RF random forest

RL reinforcement learning

RMSE root mean squared error

RNN recurrent neural network

RRC radio resource control

RRM radio resource management

RSRP reference signal received power

RSRQ reference signal received quality

RSSI received signal strength indication

RTT round-trip time

RUL remaining useful life

RX receiver

SA service and system aspects

SCS subcarrier spacing

SDF service data flow

SDU service data unit

SGD stochastic gradient descent

SINR signal-to-interference-plus-noise ratio

SIW substrate-integrated waveguide

SNR signal-to-noise-ratio

SON self-organizing network

SPS semi-persistent scheduling

SU scheduling unit

SVI stochastic variational inference

TCP transport control protocol

TR technical report

TSG technical specification group

TTI time transmission interval

TX transmitter

UE user equipment

UGRL unsupervised graph representation learning

UL uplink

UPF user plane function

URLLC ultra-reliable low latency communication

V2I vehicle-to-infrastructure

V2V vehicle-to-vehicle

V2X vehicle-to-everything

VAE-GAN variational autoencoder-generative adversarial network

VAE variational autoencoder

VI variational inference

VPN virtual private network

VT vanilla transformer

WFL wireless federated learning

WLAN wireless local area network

ZSM zero-touch network & service management

ZTN zero-touch network

cGAN conditional generative adversarial network

eNB e-NodeB

gNB gNodeB

Local k-GNN local k-dimensional GNN

pdf probability density function

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

As we approach the era of fifth-generation (5G) networks and beyond, we are witnessing an ongoing evolution toward the concept of "mobile networks for intelligence." Within the realm of 6G, a visionary landscape emerges, where networks autonomously, seamlessly, and ubiquitously exchange data, knowledge, and decision-making capabilities. This vision underscores the need for mobile networks to dynamically adapt to the diverse performance requirements of various applications. To fully harness the potential of wireless communication and address its inherent challenges, 5G is the first standard to integrate artificial intelligence (AI) into the design and management of network procedures. Accordingly, there has been a surge in research [1–9] and standardization efforts [10, 11] aimed at embedding AI into wireless systems for next-generation networks.

However, simply applying "AI for Wireless," which generally involves using narrow AI models on top of existing wireless procedures [12], is often seen as insufficient for realizing fully autonomous systems in next-generation networks [13]. A paradigm shift is required—one that transcends traditional engineering boundaries and embraces the symbiotic relationship between wireless communication and AI. The term "AI-native communications" [13–19] refers to the integration of AI technologies directly within communication networks. In this paradigm, AI is not just an add-on but an intrinsic part of the communication infrastructure, shaping how information is transmitted, processed, and managed across the entire communication system.

This Ph.D. thesis, titled "Deep Learning for Intelligent and Autonomous Wireless Networks," explores the critical intersection of wireless communication and AI, laying the foundation for the development of next-generation autonomous, AI-native communication systems. Through a holistic approach, this thesis provides a comprehensive view of the interplay between wireless communications and AI, ultimately contributing to the creation of intelligent, autonomous, and adaptive mobile radio networks.

1.1 Toward AI-Native Communication Systems

In this dissertation, I adopt a holistic perspective to analyze the evolving relationship between communication protocols and learning mechanisms. This contrasts with previous approaches where the integration of AI in communication infrastructure lacked the depth and interconnectedness I advocate here. Autonomous 6G networks will rely heavily on distributed intelligence at the network edge, where the seamless integration of communication and intelligence is essential. This thesis aims to highlight the mutual benefits of combining AI and communication systems, targeting the development of truly autonomous 6G and beyond networks, while addressing key research challenges.

The shift toward AI-native communications is driven by the heterogeneous requirements of future 6G networks, which are expected to support new services and revisit the semantic and effective communications problems introduced by Shannon's work [20]. For example, transmitting AI-related traffic for distributed training of edge devices, also known as federated learning (FL), exemplifies goal-oriented communication [21]. Traditional radio resource management (RRM) methods are generally insufficient for such applications. Thus, medium access control (MAC) procedures must evolve to allow base stations (BSs) and user equipments (UEs) to automatically learn new protocols tailored to diverse services [22]. This has led to growing research interest in fields like *learning to communicate* [23] and *protocol learning* [22, 24, 25], especially in multi-agent settings. In these interconnected systems, the goal is to learn a shared communication protocol while achieving a collaborative objective [20]. One key challenge for AI research is to extend the success of centralized algorithms to distributed systems, where entities must interact over shared wireless channels to achieve common learning objectives.

This interdependence between communication and AI opens new horizons and potential research directions, which are explored in this thesis. Section 1.3 introduces the specific research objectives addressed in this work. Another critical aspect is the seamless integration of AI into next-generation mobile networks. The following section traces the historical development of autonomous mobile radio networks, providing a conceptual foundation for this integration.

1.2 Intelligent and Autonomous Wireless Networks

This section presents a historical overview of autonomous mobile radio networks, starting from the early attempts to automate mobile networks with self-organizing networks (SONs), to the more recent emergence of the zero-touch network & service management (ZSM) group and the 3GPP's efforts to integrate AI and machine learning (ML) into RAN and service and system aspects (SA) working groups. The overarching goal of this evolution is to eliminate human intervention in network operations, leading to fully autonomous management functions.

The introduction of SON dates back to Release 8 [26, 27], with continuous improvements up to Release 17 [28]. SON focuses on automating RAN management by implementing self-configuration, self-optimization, and self-healing. From an architectural perspective, SON has traditionally been considered an overlay to the RAN, automating network parameter configuration without altering core procedures. This represents the first step toward "AI for Wireless."
The push for fully autonomous networks gained momentum in 2017 with the launch of ZSM [29]. The concept of zero-touch networks (ZTNs) introduced zero-human involvement, aiming for complete automation across all facets of network management. Unlike SON, which automates specific tasks such as configuration and optimization, ZTNs target end-to-end automation, covering everything from provisioning to maintenance. The ultimate goal is to create self-sustaining, fully autonomous networks.
Since Release 18, SON concepts have evolved into the integration of AI/ML within 3GPP RAN and SA working groups [10, 11]. AI-based automation is now considered an essential component of network design, moving beyond the role of an external oversight entity. New concepts like orchestration and intent-driven management [30–32] have emerged, enabling systems to learn and optimize network behavior in response to end-user demands. While initially applied to specific use cases, these developments reflect the growing importance of ubiquitous AI in wireless networks, aligning with the vision of standardization bodies.

1.3 Research Objectives

Achieving a fully autonomous network without human involvement is an ambitious challenge. It requires carefully identifying the key characteristics that AI must possess to meet this objective. Legitimate concerns about the AI-native design of communication techniques, ranging from the physical layer to the network layer, arise due to inherent limitations of data-driven methods. These concerns shape the research goals outlined in this thesis.
Explainability: One of the primary concerns is the opaque nature of most ML models, leading to a lack of explainability. This is particularly critical in mission-critical networks and for online optimization algorithms. Developing interpretable models with meaningful explanations is crucial.
Reliability and Trustworthiness: Deploying AI in wireless networks demands high reliability and trustworthiness to ensure stable and error-free operation. These are essential attributes for both predictive and generative models.
Robustness: Ensuring robustness, especially in the presence of outliers or rare conditions, is critical for ultra-reliable, low-latency communications. Developing calibrated probabilistic models is an active area of research [33].
Proactivity: Proactive models that can anticipate and adapt to changes in network conditions

are vital for maintaining service continuity and agile adaptation, as opposed to reactive strategies.

Adaptability: Wireless networks are dynamic and constantly evolving. AI-native systems must be capable of adapting to varying network conditions and distribution shifts. This challenge is heightened in multi-agent systems where non-stationarity and partial observability [34, 35] further complicate adaptation.

Scalability: Network optimization becomes increasingly complex as the number of elements and parameters grows. Distributed optimization in multi-agent systems can help mitigate this, but it introduces challenges like non-stationarity and partial observability.

Communication Efficiency: Efficient communication is crucial in distributed learning over wireless networks to minimize bandwidth usage and latency. Limited wireless resources demand communication-efficient protocols for effective model updates.

1.4 Thesis organization

After establishing the theoretical background necessary for the discussions that follow in this dissertation, this thesis is structured to progressively build foundational and applied insights into deep learning methods for achieving intelligent and autonomous wireless networks.

The first chapter addresses adaptive optimization within wireless networks, utilizing measurements from mobile network operators (MNOs). Within this context, model-free deep reinforcement learning (DRL) is introduced, owing to its capability to learn directly from interactions with the wireless enviroment and adjust to dynamic network conditions. To mitigate the primary limitations of model-free DRL—particularly sample inefficiency—an optimization problem is formulated, supported by a set of tailored techniques. These techniques include the development of a novel probabilistic exploration policy that enhances the agent's exploration process during training while ensuring adherence to predefined optimization constraints.

The second chapter explores the integration of generative AI in wireless networks, with the purpose of eliminating human intervention and automating complex network tasks. Incorporating generative capabilities into network systems poses unique challenges, particularly in terms of explainability, trustworthiness, reliability, and privacy. This chapter proposes an uncertainty-aware generative framework grounded in Bayesian learning to address these issues. Specifically, the framework enables the generation of synthetic mobile data from crowd-sourced data, using a conditional Bayesian-based generative model to produce mobile measurement data with high accuracy. The framework's effectiveness is evaluated through calibration analysis, interpolation capability assessment, and performance comparisons on downstream tasks using both original and synthetic datasets.

The third chapter examines distributed learning strategies in wireless networks, emphasizing

the use of graph structures to facilitate scalable, efficient distributed learning. Central to this chapter is the application of multi-agent reinforcement learning (MARL) alongside graph neural networks (GNNs) for parameterizing policies, which enable scalable optimization across networked systems. The findings highlight the advantages of employing graph-based structures as communication-inducing mechanisms within MARL, significantly improving scalability and optimization potential in wireless environments.

Finally, the fourth chapter provides insights into the design and implementation of AI-native wireless networks through a 5G industrial IoT use case. Focusing on a safety-critical scenario that leverages real sensor data, this chapter analyzes network architectures designed to estimate the remaining useful life of an AGV through deep learning. This use case illustrates how communication infrastructure affects data-driven applications in industrial IoT, underscoring the benefits, challenges, and interdependence between AI and wireless communications in next-generation networks. Overall, this thesis aims to highlight the transformative role of AI-driven techniques for autonomous wireless network management, forming a strong foundation for further advancements in intelligent and autonomous wireless networks.

Chapter 2

Deep Learning Background

This chapter provides a foundational introduction to deep learning concepts that are essential for understanding the frameworks employed throughout the thesis. Each topic is addressed to set the groundwork for the application and significance of these methods in subsequent chapters.

2.1 Reinforcement Learning

Reinforcement learning (RL) is a subfield of AI that focuses on teaching agents to map situations to actions in a way that maximizes cumulative rewards over time [36]. In RL, an agent interacts with its environment by observing its state, taking actions, and receiving feedback in the form of rewards. The agent's objective is to learn a policy that maximizes the expected cumulative reward. The two key characteristics of RL are trial-and-error search and delayed rewards [36].

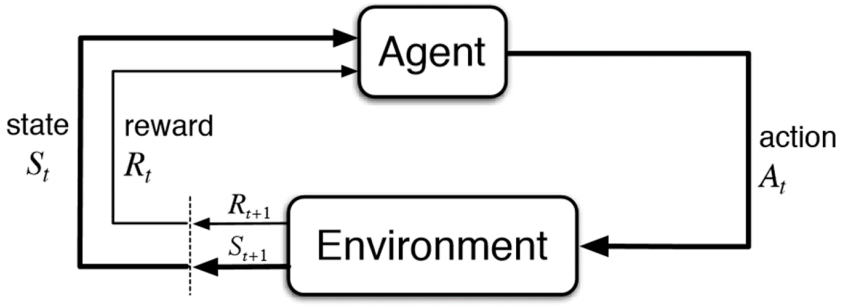


Figure 2.1: Reinforcement Learning as a block scheme: the agent aims to learn optimal behavior by interacting with the environment to obtain rewards.

Markov Decision Processes

The most common formalization of an RL problem is through a Markov decision process (MDP). An MDP provides a mathematical framework for sequential decision-making problems where outcomes are partly random and partly influenced by the agent's actions. Formally, an MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P_a, R_a, \gamma \rangle$, where:

- $\mathcal{S}$: the set of all possible states.
- $\mathcal{A}$: the set of all possible actions the agent can take.
- $P_a(s'|s,a)$: the transition probability function, which gives the probability of moving from state s to state s' given action a.
- $R(s,a,s')$: the reward function, providing the immediate reward for transitioning from state s to state s' by taking action a.
- γ: the discount factor, which weighs the importance of future rewards compared to immediate rewards. It takes values in the range $(0,1]$.

An MDP assumes the Markovian property, which means that the probability of transitioning to the next state depends only on the current state and action, not on past states or actions: $P(s_{t+1}|s_t, a_t, \ldots, s_0, a_0) = P(s_{t+1}|s_t, a_t)$.

The agent's objective is to learn a policy π, a mapping from states to actions, such that the expected cumulative reward, or return, is maximized. This objective can be mathematically expressed as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right] \tag{2.1}$$

The return can be recursively defined as the sum of the immediate reward and the discounted future reward, facilitating estimation via bootstrapping:

$$R_t = R(s_t, a_t, s_{t+1}) + \sum_{i=t+1}^{\infty} \gamma^{i-t+1} R(s_i, a_i, s_{i+1}). \tag{2.2}$$

To learn an optimal policy, the agent must balance exploring new states and actions with maximizing the reward, leading to the well-known "exploration vs. exploitation tradeoff." Various strategies have been proposed to handle this tradeoff. Section 3.5 discusses a novel approach in the context of decision-tree structured MDPs. A common collection policy for this purpose is the ε-greedy policy:

$$\pi(s) = \begin{cases} \pi^*(s) & \text{with probability } 1-\epsilon \\ \pi_{\text{random}}(s) & \text{with probability } \epsilon. \end{cases} \tag{2.3}$$

During training, the "behavior policy" often follows this ε-greedy approach, whereas, during testing, the "target policy" usually adopts a purely greedy strategy, selecting actions based solely on the learned experience.

Value-based and Policy-based Methods

In RL, two prominent approaches are value-based and policy-based methods for optimizing the agent's decision-making process.

Value-based methods focus on estimating a value function that predicts the expected return of being in a particular state or taking a specific action. One of the most widely used algorithms is Q-learning, which seeks to learn the action-value function $Q(s, a)$, representing the expected cumulative reward for taking action a in state s and following an optimal policy thereafter. The Q-values are updated iteratively using the Bellman equation:

$$Q(s, a) \leftarrow R(s, a, s') + \gamma \cdot \max_{a'} Q(s', a'). \tag{2.4}$$

For large state-action spaces, tabular Q-learning becomes computationally infeasible. Deep reinforcement learning (DRL) addresses this by using deep neural networks (DNNs) to approximate the Q-function. The deep Q-network (DQN) minimizes the temporal difference error:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(Q(s, a; \theta^{(v)}) - \left(R(s, a, s') + \gamma \max_{a'} Q(s', a'; \theta^{(t)})\right)\right)^2\right], \tag{2.5}$$

where $\theta^{(v)}$ and $\theta^{(t)}$ are the parameters of the value and target Q-networks, respectively.

Policy-based methods involve directly learning a policy $\pi(a \mid s, \theta)$, parameterized by θ, which selects actions without needing a value function. The goal is to optimize a performance measure $J(\theta)$ by adjusting the policy parameters via gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}. \tag{2.6}$$

Estimating the gradient $\widehat{\nabla J(\theta_t)}$ involves the Policy Gradient Theorem, which simplifies the computation of performance gradients by avoiding direct dependence on the unknown state distribution:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s} \mu(s) \sum_{a} Q_{\pi}(s,a) \nabla \pi(a \mid s, \boldsymbol{\theta}), \tag{2.7}$$

where $\mu(s)$ is the on-policy state distribution under π, and $Q_{\pi}(s,a)$ is the expected return of state-action pairs. Algorithms such as REINFORCE [37, 38], used in multi-agent RL problems (as discussed in Sec. 5.3), rely on Monte Carlo methods to approximate the expected reward and gradient.

2.2 Bayesian Learning

Frequentist vs. Bayesian Learning

Conventional frequentist learning focuses on identifying an optimal point estimate $\hat{\theta}$ for the parameters of a statistical model. This estimate is determined under the assumption of empirical risk minimization (ERM), where the loss function $\mathcal{L}(\theta)$, computed on the available training set, is presumed to approximate the true population loss. Optimization techniques such as minimum mean squared error (MMSE), maximum likelihood estimation (MLE), or, with regularization, the maximum a posteriori (MAP) criterion are used to estimate $\hat{\theta}$, often by employing stochastic gradient descent (SGD).

However, a fundamental issue arises from the fact that the difference between the population loss and the training loss depends on the size of the data set, which introduces uncertainty regarding the optimal parameterization. This uncertainty, termed *epistemic uncertainty* ε_{ep}, is defined as:

$$\varepsilon_{ep} = \left| \mathcal{L}(\theta^{\star}) - \mathcal{L}(\hat{\theta}) \right| , \tag{2.8}$$

where $\mathcal{L}(\theta^{\star})$ is the loss corresponding to the true optimal parameterization $\theta^{\star}$, and $\mathcal{L}(\hat{\theta})$ is the loss under the ERM assumption.

Epistemic uncertainty is generally reducible by increasing the size of the training set. In contrast, *aleatoric uncertainty* ε_{al}, inherent in the data itself, cannot be reduced by acquiring more data. By choosing a single model, frequentist learning often overlooks epistemic uncertainty, discarding valuable information about alternative models that fit the data almost as well as the ERM solution [33]. This can lead to overfitting, poor calibration, over-confident predictions in extrapolation settings, and limited explainability.

On the other hand, Bayesian learning addresses this issue by considering a distribution $\theta \sim P(\theta)$ over the model parameters. In Bayesian neural networks (BNNs), this is achieved by placing a probability distribution on the weights of the neural network instead of using fixed scalar values [39]. Each weight is assigned a posterior distribution $P(\theta \mid \mathcal{D})$, where $\mathcal{D} = \{x_i, y_i\}_{i=1}^{m}$ represents the training set.

Approximate Bayesian Methods

In Bayesian models, the predictive posterior distribution $P(y \mid x, \mathcal{D})$, which represents the distribution of predictions given the data, is obtained by marginalizing over the model parameters θ:

$$P(y \mid x, \mathcal{D}) = \int_{\theta} P(y \mid x, \theta) \cdot P(\theta \mid \mathcal{D}) \, d\theta \,, \tag{2.9}$$

where $P(y|x, \theta)$ is the likelihood under a given parameterization θ.

Since deriving this posterior in closed form is typically intractable, Eq. (2.9) is often approximated at inference time using Monte Carlo sampling:

$$P(y \mid x, \mathcal{D}) \approx \frac{1}{T} \sum_{i=1}^{T} P(y \mid x, \theta_i) \,, \tag{2.10}$$

where T is the number of samples and θ_i is the i-th sampled weight vector. This approach eliminates the need for explicit weighting in the marginalization, as less probable parameterizations are inherently sampled less frequently. By sampling from the posterior, BNNs generate multiple plausible models, each yielding slightly different predictions.

Training BNNs therefore involves approximating the posterior $P(\theta \mid \mathcal{D})$. However, computing this posterior exactly is often infeasible. To address this, approximate methods such as Monte Carlo dropout [40] and variational inference [39] have been proposed to approximate the posterior distribution.

Variational inference, for instance, involves approximating $P(\theta \mid \mathcal{D})$ with a variational distribution $Q_\lambda(\theta)$, parameterized by λ. The goal is to minimize the Kullback-Leibler (KL) divergence $\mathrm{KL}[Q_\lambda(\theta) \,\|\, P(\theta \mid \mathcal{D})]$ between the variational and true posterior distributions. This leads to the following *variational free energy* cost function [41]:

$$\arg\min_{\lambda} \; \mathrm{KL}[Q_\lambda(\theta) || P(\theta)] + E_{\theta \sim Q_\lambda}[\mathcal{L}(\mathcal{D}|\mathbf{x}, \theta)] \,, \tag{2.11}$$

which balances minimizing the loss function $\mathcal{L}(\theta)$, typically negative log-likelihood, with minimizing the complexity of the model in relation to the prior distribution $P(\theta)$.

Blundell et al. [39] approximate this cost function by applying Monte Carlo sampling during training, yielding the following tractable objective:

$$\begin{aligned} \arg\min_{\lambda} \; KL[Q_\lambda(\theta) \,\|\, P(\theta)] + \mathbb{E}_{\theta \sim Q_\lambda(\theta)}[\mathcal{L}(\theta)] &= \int_{\theta} Q_\lambda(\theta) \log \frac{Q_\lambda(\theta)}{P(\theta)} \, d\theta + \int_{\theta} Q_\lambda(\theta) \mathcal{L}(\theta) \, d\theta \\ &\approx \sum_{i=1}^{T} \log Q_\lambda(\theta_i) - \log P(\theta_i) + \mathcal{L}(\theta_i) \,. \end{aligned} \tag{2.12}$$

This optimization framework enables the effective estimation of the evidence lower bound (ELBO), an objective that balances minimizing expected training loss with reducing the divergence between the variational and prior distributions. By minimizing the objective in Eq. (2.12), the optimization process effectively balances two competing goals: reducing the expected training loss $\mathcal{L}(\theta)$, which measures how well the model fits the data under the variational distribution, and minimizing the divergence between the variational distribution $Q_\lambda(\theta)$ and the prior distribution $P(\theta)$. The latter acts as a regularization term, preventing the model from overfitting by penalizing overly complex parameterizations. This trade-off ensures that the model not only fits the data but also generalizes well by incorporating prior knowledge. This principled approach provides a robust method to quantify and incorporate epistemic uncertainty into the predictions, thereby improving reliability in generating predictions for unseen data.

2.3 Machine Learning on Graphs

Machine learning on graphs (GML) is a subfield of machine learning focused on techniques for analyzing and modeling non-Euclidean data, typically represented as graphs or networks. Graphs are versatile structures that capture complex relationships in diverse domains such as social networks, recommendation systems, biology, and, notably, wireless networks, where the topology is naturally modeled as a graph. Traditional machine learning models, such as DNNs and convolutional neural networks (CNNs), are designed to work with data in tabular formats or regular grids, making them ill-suited for the irregular structures of graph data. The central challenge in GML lies in integrating graph structure into standard machine learning frameworks. For example, in a node classification task, it is essential to incorporate information about both the node's global position in the graph and the structure of its local neighborhood [42].

A graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each edge (u, v) connects two vertices $u, v \in \mathcal{V}$ and may carry additional attributes or weights. The challenge from a machine learning perspective is to encode this high-dimensional, non-Euclidean information into a format that can be used as input for learning algorithms. Recently, many approaches have emerged to learn representations that capture structural information from graphs. These methods aim to map nodes or entire (sub)graphs into a low-dimensional vector space $\mathbb{R}^d$, where geometric relationships in the embedding space reflect the structure of the original graph. Once this embedding space is optimized, the learned representations can serve as feature inputs for downstream ML tasks.

One of the core challenges in GML is ensuring that graph representations remain invariant to permutations. That is, the representation of the entire graph should not change if the nodes are relabeled or their ordering is altered. This is captured by the notion of permutation invariance.

Mathematically, a graph function $f : \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times m} \to \mathbb{R}^d$ is permutation invariant if for any permutation $\mathbf{P} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ of the node indices, i.e., $\mathbf{P1} = \mathbf{1}$ and $\mathbf{P^T 1} = \mathbf{1}$, it holds that

$$f(\mathbf{A}, \mathbf{X}) = f(\mathbf{PAP}^T, \mathbf{PX}), \tag{2.13}$$

where $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix of the graph and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times m}$ is the matrix of node features. In addition to permutation invariance, node-level tasks on graphs require permutation equivariance. In this case, the output node representations should shift according to the permutation of the input, without altering the fundamental structure. A function $f : \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|\times m} \to \mathbb{R}^{|\mathcal{V}|\times d}$ is permutation equivariant if for any permutation $\mathbf{P}$:

$$\mathbf{P}f(\mathbf{A}, \mathbf{X}) = f(\mathbf{PAP}^T, \mathbf{PX}), \tag{2.14}$$

meaning that the node representations adapt to the node reordering, preserving the relational structure while maintaining the relative positions of nodes in the graph. These properties are crucial for generalizing across graphs of varying sizes and topologies in tasks such as node classification, link prediction, and graph classification. For an in-depth survey of recent methods for learning graph representations, readers are referred to [42]. For brevity, this section highlights a particularly successful approach for graph representation learning: deep convolutional encoders, or GNNs, which extend the convolution operation to graph structures, leveraging the powerful representation capabilities of deep learning. GNNs play a central role in this thesis, with significant applications in Sec. 5.3.

Graph Neural Networks (GNNs)

GNNs are advanced deep learning models that generate embeddings for nodes, edges, or entire (sub)graphs. The key idea behind GNNs is to learn node embeddings through the exchange of information between nodes and their local graph neighborhoods, a process known as message passing. Unlike CNNs, which use a rectangular filter for convolutions, GNNs use the graph structure as a filter, allowing them to capture both node features and the connections between nodes. This mechanism enables GNNs to approximate functions at each node while incorporating the underlying graph structure. The message-passing framework of GNNs involves two main operations: message transformation and message aggregation, which are captured by the general form of a GNN layer:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\text{AGG}\left(\{\mathbf{W}^{(l)}\mathbf{h}_u^{(l)},\ u \in \mathcal{N}(v)\}\right)\right)\ , \tag{2.15}$$

where:

- $\mathbf{h}_v^{(l+1)}$ is the updated embedding of node v in layer $l+1$.

- σ denotes the activation function.
- $\mathcal{N}(v)$ represents the neighborhood of node v.
- $\mathbf{W}^{(l)}$ is the message transformation matrix, consisting of shared weights across the graph. This weight-sharing property is fundamental to the "inductive capability" of GNNs, enabling models trained on small graphs to generalize to graphs of varying sizes.
- AGG refers to the aggregation operation, which combines messages from neighboring nodes. The aggregation must be permutation invariant/equivariant, ensuring that rearranging node inputs does not affect the output. This is critical for enabling GNNs to generalize across different nodes and various graph structures, including those in wireless networks.

Several GNN architectures have been developed, each varying in how they handle message transformation and aggregation. Two prominent examples are graph convolutional network (GCN) and GraphSAGE, described in (2.16) [43] and (2.17) [44], respectively.

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)}\mathbf{W}^{(l)}\mathbf{h}_u^{(l)}\right) . \tag{2.16}$$

In GCN, the layer aggregates information by computing a weighted average of the neighboring node features, normalized by the degree of each node.

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \mathrm{CONCAT}\left(\mathbf{h}_v^{(l)}, \mathrm{AGG}\left(\left\{\mathbf{h}_u^{(l)}, \forall u \in \mathcal{N}(v)\right\}\right)\right)\right) . \tag{2.17}$$

GraphSAGE enhances GCN by allowing flexibility in the aggregation function, as long as it satisfies permutation equivariance. It also introduces feature concatenation, combining the node's current features with the aggregated neighborhood features before applying transformations, boosting the model's expressiveness.

Chapter 3

Deep Reinforcement Learning for Radio Resource Management

This chapter explores the application of deep reinforcement learning (DRL) in the adaptive optimization of wireless communication networks, with a focus on real-world, data-driven scenarios. The increasing complexity of next-generation networks has spurred interest in automated solutions that incorporate intelligence and autonomous adaptivity into network operations [45, 46]. Reinforcement learning, particularly the model-free approach of DRL, has emerged as a promising framework for such tasks, leveraging real-time network data to dynamically adjust key parameters and optimize performance.

Adaptive optimization refers to the network's ability to proactively adjust its configuration based on current and predicted quality-of-service (QoS) levels. The ultimate goal of this approach is to enhance the performance of cellular networks by making them capable of learning from the environment, possibly predicting future network states, and responding to changing conditions in real-time. In this context, DRL offers a powerful tool for continuously refining network configurations, such as power levels, antenna parameters, and user association schemes, to meet performance objectives under dynamic and often unpredictable conditions.

In modern cellular networks, mobile network operators (MNOs) have access to large volumes of data from various sources, including key performance indicators (KPIs), drive tests, and minimization of drive tests (MDT) data [47, 48]. These data sources provide valuable information about network states and user behavior, enabling the implementation of data-driven optimization algorithms. MDT, introduced in 3GPP Release 10 [47], allows operators to collect radio measurements and location information directly from user equipment (UE). This capability reduces the need for costly manual drive tests and facilitates the aggregation of realistic network state data, which can be used to optimize network performance through data-driven approaches [49–51].

Network automation has a long history of development, beginning with the introduction of

Self-Organizing Networks (SON) by 3GPP for Long-Term Evolution (LTE) [52, 53]. SON was designed to reduce manual intervention in the planning, configuration, management, and optimization of mobile networks. With the advent of more complex network architectures and growing demands for higher performance and lower operational costs, the evolution towards a zero-touch paradigm has begun, where artificial intelligence (AI) plays a central role in minimizing human involvement in network operations. DRL, in particular, is well-suited for this role, as it can autonomously learn optimal control policies for radio resource management tasks by interacting with the environment and continuously improving based on feedback.

The shift from model-driven to data-driven approaches in AI has been facilitated by the growing availability of data and computational resources [54]. Data-driven methods, such as those powered by DRL, are particularly advantageous in the context of radio resource management, as they can exploit large datasets to optimize network performance more effectively than traditional, model-based methods. For example, DRL-based algorithms have been applied to capacity and coverage optimization (CCO), where network parameters are adjusted to maximize both signal coverage and user throughput [55].

In this chapter, we explore adaptive optimization using DRL, where the network continuously learns from network-level data and user behavior to adjust its operational parameters. As an illustrative example, we focus on the challenge of optimizing both capacity and coverage in mobile radio networks. CCO is a widely studied Pareto optimization problem under the framework of SON [46, 56–59], with particular relevance to delivering predictive quality of service (PQoS) in mobile networks. One common method for performing CCO is through dynamic adjustments of coverage areas by fine-tuning power levels or antenna parameters. This method has a threefold effect: (i) mitigating interference, (ii) balancing mobility load, and (iii) adjusting coverage conditions, particularly at the network edge.

Achieving adaptability in the context of CCO is a formidable challenge, as it requires modeling and understanding the complex dynamics of mobile radio networks. In this regard, model-free DRL has gained increasing attention due to its ability to operate without prior knowledge of the environment's transition dynamics or reward structures. This feature makes model-free DRL especially valuable in scenarios where accurately modeling the environment is either difficult or computationally expensive, as it learns through a process of trial and error.

To address the need for adaptability, we propose leveraging a continuous data collection framework, combined with DRL techniques, as a powerful tool. Specifically, this chapter introduces a framework that integrates diverse network key performance indicators (KPIs), minimization of drive test (MDT) data, and electromagnetic planning tools with model-free DRL. This approach is essential for achieving the adaptability required for CCO. We further demonstrate how DRL can address key challenges in radio resource management, such as antenna down-tilt tuning for CCO, and we emphasize the potential of MDT-driven DRL solutions for real-world cellular

networks [49]. In addition, we review the literature on data-driven algorithms for network automation, evaluating their effectiveness in meeting the increasing complexity of next-generation networks.

3.1 Literature overview

CCO is a challenging optimization problem in cellular networks that has been extensively studied, particularly within the broader scope of SON [46, 56, 57]. One common approach to address CCO is to adjust the coverage area by modifying the tilt of antenna elements. Earlier contributions from the past decade predominantly employed classical operations research methods [56, 57] to fine-tune antenna parameters. However, in recent years, driven by the success of deep learning and RL, DRL algorithms have gained traction as effective tools for solving CCO [58–61]. RL is particularly well-suited for CCO because of its ability to learn and adapt to environmental dynamics [58]. For instance, in [58], a two-step algorithm that combines multi-agent mean-field RL with single-agent RL is proposed as a scalable solution for online antenna tuning in multi-tier networks. In [59], the authors address the CCO problem by using deep deterministic policy gradient (DDPG) and Bayesian optimization to balance under- and over-coverage. Additionally, [61] tackles the issue of unstable hyper-parameter convergence when optimizing antenna parameters in heterogeneous network (HetNet) environments. Finally, [60] introduces a safe RL-based policy for antenna tilt updates, designed to prevent performance degradation by avoiding harmful actions, thus improving system reliability. These studies represent significant advancements, offering solutions to enhance scalability [58, 61], sample efficiency [59], and reliability [60]. However, none of them utilize real-world data, such as MDT measurements, for network state representation and model training. The closest approach is found in [59], where electromagnetic simulations are used to generate reference signal received power (RSRP) maps. In this chapter, we propose a DRL agent trained through direct interaction with a simulated network environment, leveraging MDT data, network KPIs, and electromagnetic simulations provided by a network operator.

3.2 System Model

The system model considers a cellular network deployment within a designated area of interest, comprising both “target” and “boundary” base stations. The target base stations represent those actively optimized by the DRL agent, while the boundary base stations, although not directly optimized, are included to account for the effects at the network’s edges that may arise from the agent’s actions. The reference scenario focuses on the deployment of an MNO’s network within an approximately 26 [km^2] area located in the northern region of Bologna, Italy (see Fig.

3.1). This deployment includes a total of 18 base stations, with 9 classified as “target nodes” (depicted by pink markers in Fig. 3.1), and the remaining ones, shown in grey, categorized as “boundary” cells.
The selected area presents a particularly challenging network configuration due to its heterogeneous propagation characteristics, which encompass highways, urban areas, and agricultural fields.

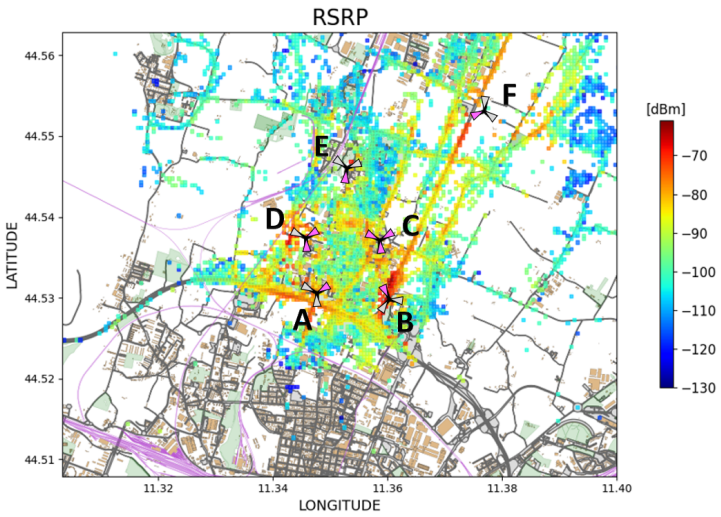


Figure 3.1: Network deployment - North Bologna Area.

Within this designated area, an offline approach is adopted for network parameter optimization. Specifically, data collected from this region is used to develop a network simulator, which serves as the training environment for the DRL agent. In Fig. 3.2, the system is illustrated as a block diagram: the simulated network environment receives input in the form of MDT data, traffic KPIs, and electromagnetic simulations. The environment is designed to accurately represent the impact of parameter reconfigurations (e.g., antenna tilt adjustments), which correspond to the actions taken by the agent, on overall network performance. A set of data processing methods is employed within the environment to ensure realistic simulations, as detailed in the following sections.

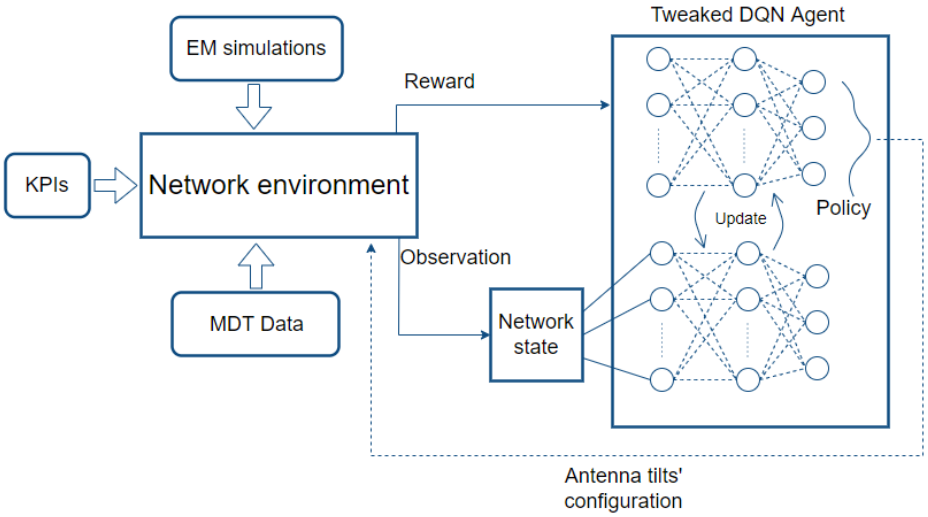


Figure 3.2: System Model - A DRL agent interacts with a simulated network environment, receiving as input network KPIs, MDT data, and electromagnetic simulations.

3.3 Simulated Network Environment

The simulated network environment is made up of four logical blocks performing different tasks, as shown in Fig. 3.3.

Pre-processing

The MDT data is divided into time slots characterized by quasi-static traffic behavior. This behavior can be analyzed using the intensity measure $\nu(A)$, which represents the expected number of samples in the Poisson point process (PPP) of MDT samples over a region A. The intensity measure is given by:

$$\nu(A) = \int_{A \subset \mathbb{R}^n} \beta(\mathbf{s})\, d\mathbf{s}\,, \tag{3.1}$$

where $\beta(\mathbf{s})$ is the intensity function that varies with spatial coordinates.
The MDT data are then aggregated into pixels of fixed size to reduce the variance of individual measurements and ensure consistent input dimensions for the DQN. The pixel size is determined based on $\beta(\mathbf{s})$ to balance the trade-off between quantization error and the scarcity of reports per pixel. Each pixel is characterized by several key quantities, which are evaluated as follows:

1. **RSRP**: This represents the linear average of individual MDT RSRP measurements.

2. **WEIGHT**: At the start of each training episode, each pixel is assigned a weight sampled

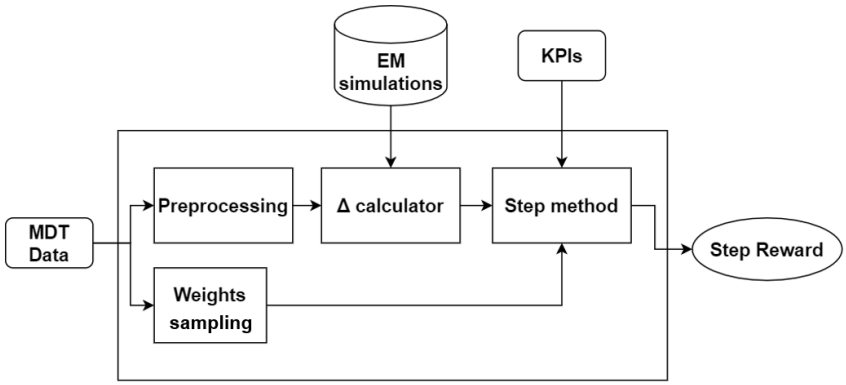


Figure 3.3: Simulated environment - block system view

from a Poisson distribution (see the weights sampling block in Fig. 3.3). The rate parameter λ of the distribution is set to the number of RRC connection establishments within the pixel. Since the timing of each RRC connection may vary, the number of reports generated by each UE can differ significantly depending on the duration of their connection. By assigning weights based on the number of RRC connections relative to the total number of MDT samples, this method ensures a fair representation of users and prevents overfitting to the training data.

3. **SINR**: The signal-to-noise-and-interference ratio (SINR), denoted as γ, is computed from individual reference signal received quality (RSRQ) measurements using the following equation:

$$\gamma = \frac{12 \cdot \text{RSRQ}}{1 - \rho \cdot \text{RSRQ} \cdot 12} , \tag{3.2}$$

where RSRQ is defined as [62]:

$$\text{RSRQ} = \frac{N_{\text{PRB}} \cdot \text{RSRP}}{\text{RSSI}} , \tag{3.3}$$

with N_{PRB} representing the number of resource blocks in the E-UTRA carrier's reference signal strength indicator (RSSI) measurement bandwidth.

In (3.2), ρ denotes the percentage of occupied physical resource blocks (PRBs) in the serving cell, and 12 refers to the number of OFDM sub-carriers in a PRB. The numerator, which represents the aggregated RSRQ on a pixel basis, reflects the serving cell's co-channel contribution, as RSRP measures the power of the demodulated channel reference signal (CRS) symbols. The denominator, representing RSSI, is a wideband measure of co-channel serving and non-serving cells, adjacent channel interference, and noise. As a result, (3.2) offers a more accurate description of the signal-to-noise-and-interference ratio when performance is limited by interference and load variations in cells, compared

to the conventional power ratio formulation.

Electromagnetic Simulator

Electromagnetic simulations are performed using proprietary software provided by a MNO. These simulations are stored for every cell and each selectable antenna parameter, resulting in a total of $P \times C$ simulations, where P is the number of parameter configurations and C is the number of cells in the cluster. Each simulation computes the electric field intensity (measured in [dBuV/m]) on a pixel basis, considering each cell as the emitting source individually. The electric field intensity is then converted to the RSRP (measured in [dBmW]) by assuming a reference receiver bandwidth of one resource element (RE) (15 [kHz]). Each pixel is assigned to a primary serving cell based on a best-server criterion. From the RSRP received from various emitting sources, the SINR, denoted as γ, can be computed for each pixel as the ratio of useful power to interfering power. In this context, each pixel is concurrently associated with both MDT-based RSRP and γ values as well as their simulated counterparts. This allows for the computation of two metrics, Δ_R and Δ_γ, which represent the differences (in dB) between the simulated RSRP and γ values and the corresponding MDT-derived measurements for each pixel. These metrics are computed by the "Δ calculator" module shown in Fig. 3.3 and reflect the estimation error of the electromagnetic simulation tool compared to the measurement reports from individual UEs. This estimation error is assumed to be independent of the antenna configuration.

"Step" Method

The step method simulates the impact of an agent's action on network performance, given a state observation. It returns the immediate reward, R_{t+1}, and the next state, S_{t+1}, as output. Formally, the method executes the transition probability matrix of the MDP associated with the environment, expressed as:

$$\Pi = P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, \pi(S_t) = a), \quad \text{for } a \in \mathcal{A}\,. \tag{3.4}$$

In (3.4), $\pi(S)$ represents the policy followed by the DRL agent, which varies based on the algorithm used (e.g., value-based vs. policy-based, Sec. 2.1). Since modeling Π directly is typically infeasible due to the high dimensionality of the action-state space, the method defines a series of sequential operations to return the tuple $\{R_{t+1}, S_{t+1}\}$, based on the current state and action $\{S_t, A_t = \pi(S_t)\}$. This process draws from the building blocks shown in Fig. 3.3 to modify the current network state and calculate the reward associated with the agent's action. The method performs the following procedures in each iteration:

1. **Weight initialization**: Weights are sampled from a Poisson distribution, as described in Sec. 3.3.

2. **User distribution**: The number of UEs in the RRC_connected state, referred to as "act_UEs," is gathered for each cell during a specific time slot from the input network KPIs. This total count is then redistributed among pixels based on their weights, associating each pixel with a percentage of the total UEs described by the KPI.

3. **Simulated KPIs**: Based on the action selected by the agent's policy $\pi(S_t)$, the simulated RSRP and γ values are computed for each pixel.

4. **Cell reselection**: Each pixel's simulated RSRP is evaluated according to the best-server criterion:
$$\max\left[\text{RSRP}_1, \text{RSRP}_2, \ldots, \text{RSRP}_C\right] \ .$$
The serving cell is reassigned accordingly.

5. Δ **Correction**: A Δ correction is applied to the simulated RSRP and γ values using:
$$\begin{aligned} \text{RSRP}'_i &= \text{RSRP}_i + \Delta_R \ , \\ \gamma'_i &= \gamma_i + \Delta_\gamma \ . \end{aligned} \tag{3.5}$$
This correction accounts for estimation errors in the electromagnetic simulations, preserving the valuable information in the MDT data, including γ computed as a function of RSRQ and network load ρ.

6. **Recompute user distribution**: Based on the new pixel assignments (step 4), the number of "act_UEs" in each cell is recomputed according to the proportion of active UEs within each pixel.

7. **Reward calculation**: The immediate reward R_{t+1} is computed based on the new state S_{t+1}.

3.4 Markov Decision Process Formulation

The predominant approach for formalizing an RL problem is to represent it as a Markov decision process (MDP). This section introduces the action space, state space, and reward function, and outlines the corresponding optimization problem.

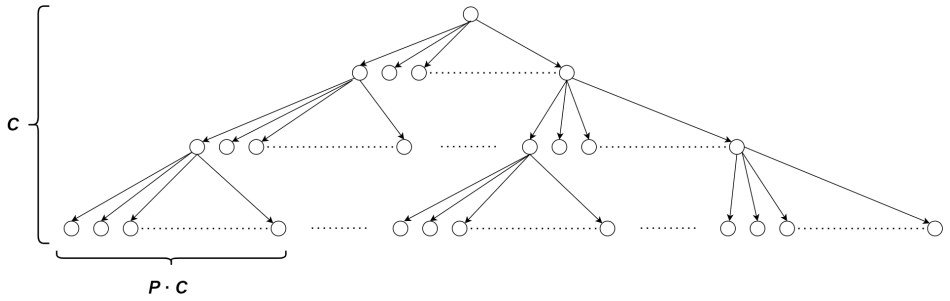


Figure 3.4: MDP formulation as a decision tree

Action Space

The centralized agent operates within an exponentially growing action set, determined by a discrete set of optimization parameters denoted as $\mathcal{P}$, with cardinality $|\mathcal{P}| = P$, across a set of $\mathcal{C}$ cells ($|\mathcal{C}| = C$). The complexity of this formulation grows exponentially, $\mathcal{O}(P^C)$, making it intractable for medium to large clusters. To overcome this challenge, the optimization problem is more effectively framed as a sequential learning task, where the MDP is structured as a decision tree.

At each step of the episode, the agent selects both a target cell from the available set and an appropriate configuration of its parameters. This approach reduces the action space to $P \times C$ at each step (Fig. 3.4). However, this simplification may yield suboptimal solutions, requiring techniques for efficient exploration and pruning of the decision tree, discussed in the next section.

Although the total number of nodes in the decision tree grows as $(P \times C)^{C+1} - 1$, this MDP formulation enables sequential pruning based on estimated Q-values during training. This approach introduces the classic tradeoff between exploitation and exploration. Exploiting known state-space knowledge (i.e., Q-value estimates) facilitates faster pruning by discarding non-promising branches. Conversely, prolonged exploration enables a more thorough search of the state space, reducing the risk of suboptimal, locally optimal solutions.

Thus, the action space $\mathcal{A}$ is defined as:

$$\mathcal{A} = \mathbf{P}_{P\times C} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,C} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ p_{P,1} & p_{P,2} & \cdots & p_{P,C} \end{bmatrix}, \tag{3.6}$$

$$\text{with } p_{i,j} \in \{0,1\} \text{ and } \sum_{i=1}^{P}\sum_{j=1}^{C} p_{i,j} = 1\ ,$$

where $\mathcal{P}$ ($|\mathcal{P}| = P$) represents the set of antenna parameter configurations (5 in this case), and

$\mathcal{C}$ ($|\mathcal{C}| = C$) represents the set of target base stations (9 in this instance), with $p_{i,j}$ denoting a specific antenna configuration for each base station.

State Space

At each time step, the DQN agent receives a network state observation, $\mathcal{S}$, described by the following quantities:

$$\mathcal{S} = \begin{cases} \mathbf{R}_{m\times n} & \text{with } r_{i,j} \in [-140, -40] \\ \mathbf{W}_{m\times n} & \text{with } w_{i,j} \in \mathbb{R}^{+} \\ \boldsymbol{\gamma}_{m\times n} & \text{with } \gamma_{i,j} \in \mathbb{R} \\ \mathbf{h} & \text{with } h_i \in \{0,1\},\ |\mathbf{h}| = P \times C\ , \end{cases} \tag{3.7}$$

where $\mathbf{R}_{m\times n}$, $\mathbf{W}_{m\times n}$, and $\boldsymbol{\gamma}_{m\times n}$ are matrices of dimensions $m \times n$. Each element $r_{i,j}$, $w_{i,j}$, and $\gamma_{i,j}$ represents the RSRP, weight, and γ values for a pixel with coordinates (i, j). These quantities are essential for addressing the multi-objective nature of CCO problems:

- $r_{i,j}$ helps manage mobility and evaluate network coverage.
- $\gamma_{i,j}$ determines link spectral efficiency and user capacity.
- $w_{i,j}$ distinguishes between more and less relevant pixels in terms of traffic load.

Additionally, a vector $\mathbf{h}$, representing the episode's history of actions using one-hot encoding, is included in the state space to account for the agent's memory of past actions during an episode.

Optimization Problem

The reward function for a CCO problem should reflect its multi-objective nature, balancing maximization of average end-user capacity with coverage conditions. The optimization problem is thus formulated as follows:

$$\text{Maximize} \quad u(\mathbf{P}) \tag{3.8}$$

$$\text{Subject to} \quad k(\mathbf{P}) \geq \Gamma \tag{3.8a}$$

$$\sum_{i=1}^{P} p_{i,j} = 1,\ \text{for } j \in \{1, \ldots, C\}\ . \tag{3.8b}$$

In (3.8), the objective function $u(\mathbf{P})$ represents the average end-user throughput as a function of the antenna configuration parameters $\mathbf{P}$ and is defined by:

$$u(\mathbf{P}) = \sum_{j=1}^{C} \frac{N_j}{N} \left(\frac{B_j}{\sum_{i=1}^{L_j} w_{i,j}} \sum_{i=1}^{L_j} w_{i,j}\, \eta_{i,j}(\mathbf{P}) \right) , \tag{3.9}$$

where N_j is the number of active UEs in the j-th cell (step 6, Sec. 3.3), L_j is the number of pixels, and B_j represents the bandwidth per user. Here, $w_{i,j}$ and $\eta_{i,j}$ represent the weight and link spectral efficiency, respectively, for the i-th pixel in the j-th cell. The link spectral efficiency $\eta_{i,j}$ is computed using channel state information and modeled as a function of the signal-to-interference-plus-noise ratio (SINR), thereby capturing both frequency-selective fading and interference effects. In practice, $\eta_{i,j}$ is obtained by averaging the spectral efficiency over the frequency slots allocated to each user, accounting for variations in attenuation and interference across different frequencies.

The optimization problem in (3.8) is subject to two constraints: (3.8a), which ensures that the coverage function $k(\mathbf{P})$ exceeds a threshold Γ pixels, and (3.8b), which ensures that the sum of the action elements in each column of $\mathbf{P}$ equals 1, implying that only one parameter configuration is chosen for each target cell in $\mathcal{C}$.

The coverage constraint $k(\mathbf{P})$ is defined as:

$$k(\mathbf{P}) = |\{(i,j) \mid r_{i,j}(\mathbf{P}) \geq r^*,\ i \in \{1,\dots,L_j\},\ j \in \mathcal{C}\}| \ , \tag{3.10}$$

which counts the number of pixels (i,j) where the RSRP value $r_{i,j}$ exceeds the threshold r^*.

To solve the optimization problem with DRL, the dual Lagrangian approach is employed, incorporating unitary multipliers to relax constraints (3.8a) and (3.8b). To explicitly model the dependency on Γ, we redefine (3.10) as:

$$k(\mathbf{P}, \Gamma) \propto \begin{cases} e^{\Gamma - k(\mathbf{P})}, & \text{if } k(\mathbf{P}) \leq \Gamma \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

By incorporating a Heaviside step function $h(\mathbf{P})$, which introduces a constant penalty depending on whether constraint (3.8b) is satisfied, the objective function (3.8) can be rewritten as:

$$\arg\min_{\mathbf{P}} -u(\mathbf{P}) + k(\mathbf{P}, \Gamma) + h(\mathbf{P}) \ . \tag{3.12}$$

3.5 Sample-efficient Deep Reinforcement Learning

One of the primary challenges of model-free DRL is its sample inefficiency. To enable fast adaptation to dynamic scenarios, it is crucial to develop a training paradigm that improves sample efficiency. This section presents a detailed summary of the optimization strategies designed

to address this issue. Specifically, a novel exploration policy, the "depth-wise $\epsilon - \eta$-greedy policy," is introduced to enhance sample efficiency significantly.

1. **Decision Tree**
   The first enhancement involves reformulating the MDP as a decision tree, which allows the agent to prune unpromising branches and achieve faster convergence. This formulation, however, introduces the challenge of efficiently balancing the exploration-exploitation tradeoff.

2. **Episode History**
   Integrating episode history directly into the state observation enables the agent to leverage memory of its past actions throughout the episode. This approach facilitates learning an action set that satisfies constraint (3.8b).

3. **Lagrangian Relaxation**
   By relaxing constraints, the agent can explore the state-action space more effectively, accelerating convergence.

4. **Depth-wise $\epsilon - \eta$ Greedy Policy**
   This custom exploration policy, combined with Lagrangian relaxation, extends the traditional ϵ-greedy policy for environments requiring complex action patterns and events that occur with sparse probabilities. The depth-wise $\epsilon - \eta$-greedy policy introduces a probabilistic approach that guides the agent's exploration phase by enforcing soft constraints during training. It introduces an additional parameter, η, which controls the likelihood of performing a "constrained random action," i.e., a random action sampled from a set of constraint-compliant actions. The pseudo-code of the proposed method is provided in Algorithm 1.

For the given optimization problem, the goal is to optimize a different cell at each step of the episode. The policy in Algorithm 1 helps enforce constraint (3.8b) with a probability controlled by the parameter η (Eq. (3.16)) at every episode step. This is important because the probability of randomly completing a constraint-compliant episode decreases exponentially with the number of steps. Denoting $X^{(i)}$ as a random variable representing the probability of completing the i-th episode step in a constraint-compliant manner, the probability $P(X)$ of completing a constraint-compliant episode is given by:

$$P(X) = P(X^{(C)}, X^{(C-1)}, \ldots, X^{(1)}) \underbrace{=}_{(a)} \prod_{i=0}^{C-1} \left(1 - \frac{i}{C}\right) = \frac{(C-1)!}{C^{C-1}}\,, \tag{3.13}$$

where (a) follows from the chain rule of probability, and the number of episode steps equals the number of cells C in the cluster. Since $(C-1)! < C^{C-1}$, the inverse of the probability grows

Algorithm 1 Depth-wise $\epsilon - \eta$ Greedy Policy

$i \leftarrow$ random(0, 1)
$j \leftarrow$ random(0, 1)
$\epsilon \leftarrow \epsilon$_scheduling
$\eta \leftarrow \eta$_scheduling
if $i \geq \epsilon$ **then**:
    perform greedy action
else
    if $j \geq \eta$ **then**:
        perform constrained random action
    else
        perform random action
    end if
end if

rapidly as the problem size increases. Hence, a purely random exploration strategy (ϵ-greedy) becomes inefficient, as only one episode out of $\frac{(C-1)!}{C^{C-1}}$ will, on average, complete without penalties. By controlling the η parameter for selecting actions from a constraint-compliant set, we can balance positive rewards and penalties, thereby mitigating the issue of sparse rewards. Importantly, this policy does not entirely prevent the agent from choosing an action that violates a constraint (unless $\eta = 1$), as required by the Lagrangian optimization.
The practical benefits of introducing the η parameter, confirmed by numerical experiments, are twofold:

1. The exploration phase duration is significantly reduced.

2. The training process becomes more stable.

Scheduling of the Exploration Parameters

The decision-tree MDP formulation enables pruning based on estimated Q-values, allowing the agent to focus on promising branches while discarding large portions of the state-action space. As Q-value estimates improve over successive training episodes, the ϵ parameter is typically scheduled to decrease monotonically, with exploration gradually giving way to exploitation as the agent gains knowledge from interacting with the environment.
Given that the number of nodes in the tree increases exponentially with depth, the first-level nodes are encountered much more frequently than the leaf nodes. Therefore, using a single scheduling function for the ϵ parameter across all levels may not yield optimal results. A better approach involves assigning a distinct scheduling function to each level, where the ϵ decay rate is faster at earlier levels to allow for sequential pruning of the tree. This depth-wise scheduling strategy for ϵ is illustrated in Fig. 3.5.

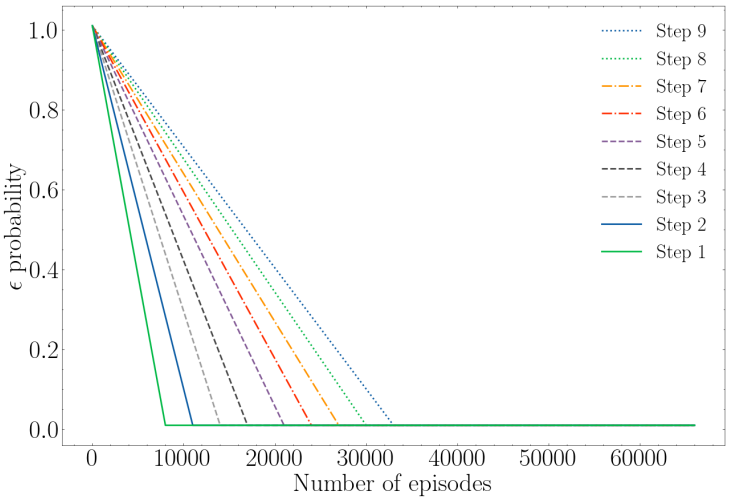


Figure 3.5: Depth-wise ϵ scheduling

Similarly, scheduling the η parameter based on the tree depth is beneficial. Since the probability of complying with constraint (3.8b) varies with depth, it makes sense to use higher η values for deeper steps. The goal is to distribute positive and negative experiences evenly across all levels of the tree. This can be represented in terms of probabilities as:

$$P(X^{(i)}) = P(X^{(i)}|\,\bar{\eta})(1-\eta) + \underbrace{P(X^{(i)}\,|\eta)}_{=1}\,\eta = P(X^{(i)}|\,\bar{\eta})(1-\eta) + \eta, \tag{3.14}$$

$$P(X) = P(X^{(C)}|X^{(C-1)}, \ldots, X^{(1)}) = \prod_{i=1}^{C} P(X^{(i)}), \tag{3.15}$$

where $P(\bar{\eta}) = (1-\eta)$ is the probability of taking a random action, and $P(\eta) = \eta$ is the probability of selecting a constraint-compliant action. From Eqs. (3.14) and (3.15), we can derive:

$$\eta = \frac{P(X)^{1/(C-1)} - P(X^{(i)}|\,\bar{\eta})}{1 - P(X^{(i)}|\,\bar{\eta})}. \tag{3.16}$$

Table 3.1 presents the η values for each depth level when $P(X) = 0.5$.

DQN Architecture

The optimization techniques described above are implemented using a deep Q-network (DQN) algorithm [63]. The DQN architecture employed in training is depicted in Fig. 3.6. Input data consists of MDT pixels from the pre-processing block (Sec. 3.3) and episode history. Two separate branches of the DQN process these streams before concatenating and feeding them

Decision Tree Depth	$P(X^{(i)}\mid \bar{\eta})$	η
1	1	0
2	8/9	0.253
3	7/9	0.627
4	6/9	0.751
5	5/9	0.813
6	4/9	0.851
7	3/9	0.876
8	2/9	0.893
9	1/9	0.907

Table 3.1: Depth-wise η scheduling

into a shared fully connected layer. The first branch, processing MDT pixels, comprises one 5x5 2D convolutional layer followed by three fully connected layers, without pooling layers to preserve spatial information. The second branch, handling episode history, directly feeds into the concatenate block. A fixed Q-targets variant of the DQN algorithm is used, with separate target and action networks updated every 1500 training steps.

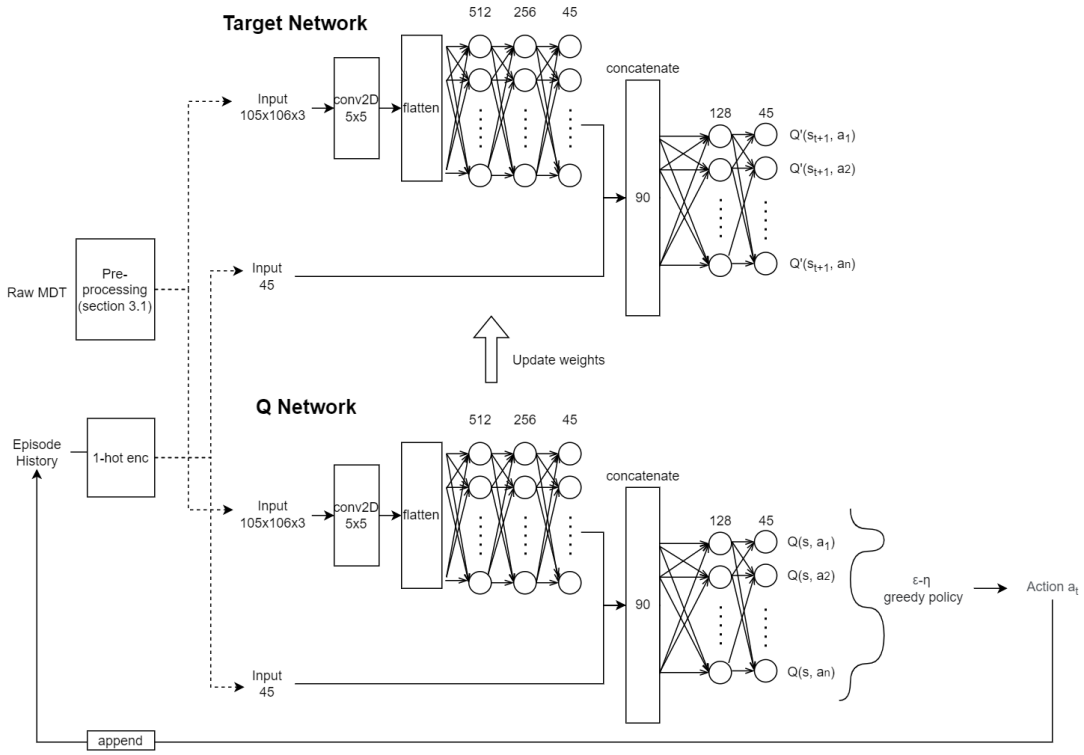


Figure 3.6: Deep Q-Network architecture

3.6 Experimental Results

Numerical results were obtained by comparing the proposed enhanced DQN formulation against several baseline algorithms, as described below:

- **Best First Search (BFS)**: A tree exploration algorithm that exhaustively explores each branch of the decision tree up to a depth of one level. At each step in an episode, BFS explores every branch using a brute-force approach and selects the best branch as the source node for the next iteration. By prioritizing the most promising branches, BFS helps in finding solutions more efficiently.

- **Vanilla DQN**: A standard DQN implementation that uses an ϵ-greedy exploration policy.

The results demonstrate significant improvements in three key metrics: (i) episode reward, (ii) sample efficiency, and (iii) training stability compared to the baseline methods.
Fig. 3.7 presents the training curves of the proposed depth-wise $\epsilon\eta$-greedy DQN in comparison to the vanilla ϵ-greedy DQN. The training curves were generated by evaluating the greedy policies of both algorithms using 10 random seeds at every 1000 episode steps during training. As shown in the figure, the proposed method shows a marked improvement in performance compared to the baseline. Specifically, the depth-wise $\epsilon\eta$-greedy DQN exhibits greater stability,

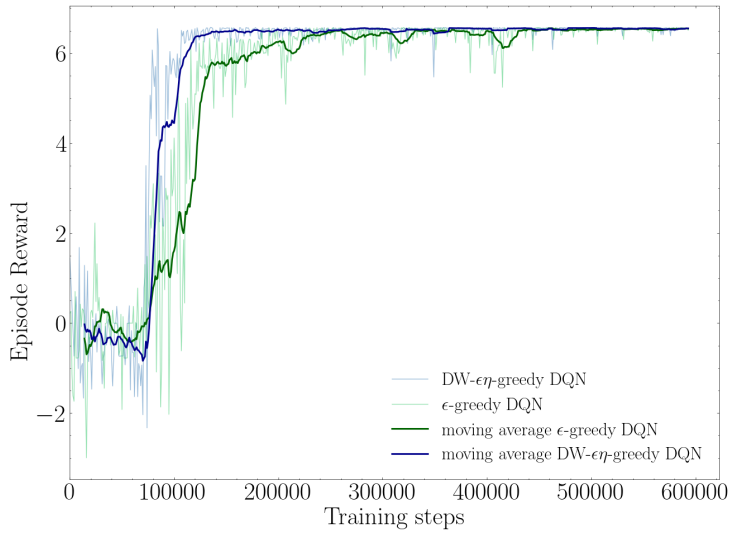


Figure 3.7: Training curves: Depth-wise $\epsilon\eta$-greedy DQN vs. ϵ-greedy DQN.

slightly improved performance, and significantly enhanced sample efficiency, reaching a stable plateau 70% faster (Fig. 3.7a).

Fig. 3.8 illustrates the reward obtained at each episode step by the three algorithms. The results were derived from a Monte Carlo experiment and are shown with 99% confidence intervals. Both the ϵ-greedy DQN and depth-wise $\epsilon\eta$-greedy DQN efficiently learn to forgo immediate rewards in favor of better cumulative episode rewards. Notably, the depth-wise policy of the $\epsilon\eta$-greedy DQN demonstrates an interesting behavior: its policy is more compact compared to the ϵ-greedy DQN, as reflected by the narrower confidence intervals. This leads to fewer reconfigurations of network parameters, with more environment states being mapped to the same antenna configurations.

In RL, it is common to observe significant variability between different runs, where small changes, such as different random seeds, can result in distinct statistical distributions [64]. This variability highlights the importance of sensitivity to minor fluctuations in the environment when assessing the performance of an RL algorithm. Fig. 3.9 illustrates this phenomenon using boxplots that display the distribution of episode rewards for the three algorithms across 50 runs, each using a different random seed. The depth-wise $\epsilon\eta$-greedy DQN not only outperforms the baseline algorithms in terms of average episode reward, but it also shows a variance level similar to that of BFS. It is worth noting that BFS, as an informed search algorithm, is less affected by the inherent variability of RL.

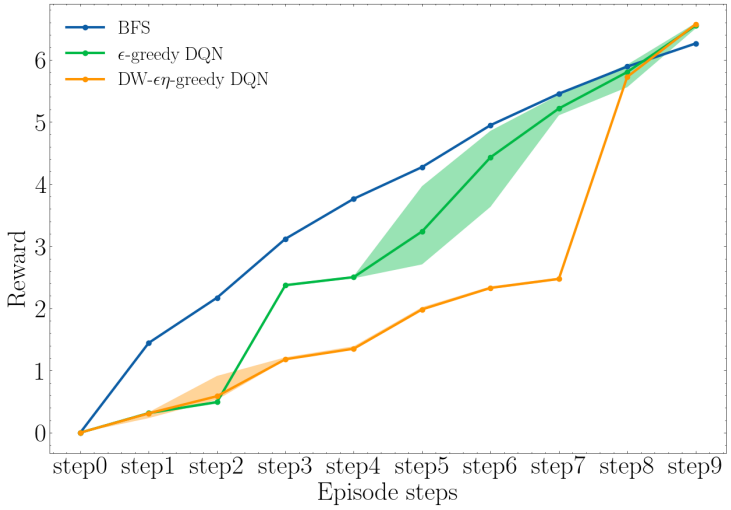


Figure 3.8: Step reward comparison: Depth-wise $\epsilon\eta$-greedy DQN vs. ϵ-greedy DQN vs. BFS.

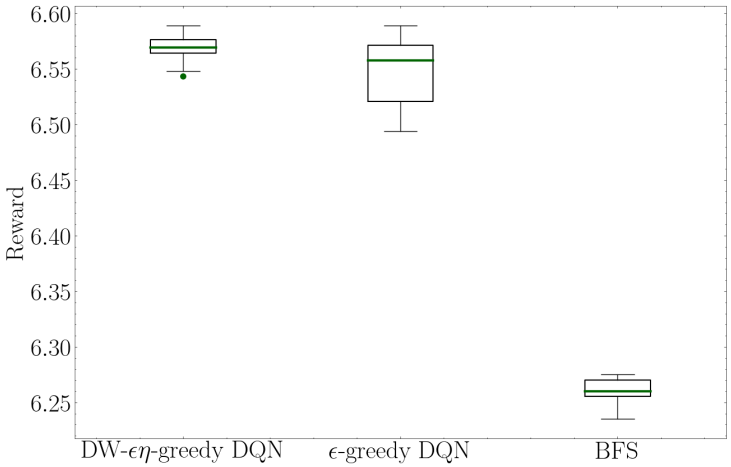


Figure 3.9: Average episode reward distribution across 50 runs.

Chapter 4

Bayesian Learning for Data Generation in Wireless Networks

4.1 Motivations and Challenges

Generative artificial intelligence (GAI) is a sub-field of AI that focuses on creating models capable of generating new data, such as images, text, or audio [65]. It has emerged as a transformative technology, driving revolutionary advances across various domains, including healthcare, arts, computer vision, natural language processing, industry applications, and notably, wireless communications and networking.

The application of GAI in enabling autonomous and intelligent wireless systems is an area of active and rapidly evolving research. By leveraging its ability to model complex distributions and simulate network behaviors, GAI can be used to construct generative models and digital twins (DTs) that learn from real-world data and produce synthetic samples resembling the original data [66]. This is particularly valuable when access to real data is limited or when data collection is challenging [65, 67, 68]. Furthermore, this generative ability enhances the development of systems that can *generalize* and *adapt* beyond local observations.

However, integrating generative capabilities into wireless systems to minimize human intervention brings several significant challenges. These include concerns about *explainability*, *trustworthiness*, *reliability*, and *privacy*. The opacity of many GAI models often limits our understanding of their decision-making processes, which is critical in wireless networks, particularly for mission-critical applications where the reasoning behind decisions must be clear. Building trust in these models requires thorough validation, verification, and testing to avoid risks from biased or incorrect decisions. A promising solution to these challenges lies in the use of uncertainty-aware generative processes, which can be developed with the theoretical assurances provided by Bayesian Learning, as discussed further in Sec. 4.3.

In this section, we explore how GAI can be employed for the synthetic generation of mobile

data. Within this framework, GAI can be seamlessly integrated into the PQoS model to develop DTs, thereby enhancing predictive capabilities and enabling more efficient and rapid adaptation of network resources.

4.2 Literature Overview

Although GAI has seen its most significant success in fields such as computer vision [69] and natural language processing [70], its application to communications and networking [68, 71] is an emerging and active area of research. This field has found numerous practical applications, ranging from physical layer modeling to network management. Many works in this domain focus on the use of generative adversarial networks (GANs) [72–74] for generating synthetic datasets.

For example, GANs have been employed to learn probabilistic channel distributions [75–77], which enables accurate channel modeling for various downstream tasks, such as multiple-input multiple-output (MIMO) precoder design. The application of GANs has also been widely explored for generating mobile traffic data and radio maps. Notably, in [78], GANs are used to augment datasets consisting of call data records (CDRs), which are tabular records detailing the average start time and duration of phone calls in mobile networks. The augmented dataset enhances the predictive accuracy of an autoregressive task by leveraging the additional synthetic data generated by the GAN framework.

The use of CDRs data is further expanded in [79], where Di Paolo et al. introduce a comprehensive framework for assembling an extensive dataset aimed at network planning. This framework models distributions from diverse data sources, including CDRs, demographic information, and network deployment details obtained from MNOs. In [80], Sun et al. present a deep generative framework capable of producing synthetic time-series data for unseen trajectories during training. The framework generalizes by abstracting information from network and environmental contexts, such as cell site locations, estimated transmit power, cell orientation, and environmental factors like terrain, obstacles, and clutter. Similarly, [81] proposes a method for predicting signal quality metrics in long-term evolution (LTE) networks at unobserved locations, using raw GPS measurements, network context (e.g., distance to transmitters), and satellite images. To improve radio map estimates, [82] introduces an innovative approach that combines radio propagation models with a conditional generative adversarial network (cGAN) architecture.

Moreover, trained GANs can reconstruct high-dimensional data from low-dimensional inputs with fewer generator function constraints than other models. This makes them particularly effective for physical layer communication tasks such as channel estimation [83], channel state information (CSI) compression [84], and physical layer security [85].

Lastly, GAI plays a key role in the internet-of-things (IoT) domain, particularly in generating

data for privacy-sensitive [86] or sparsely populated datasets [87, 88]. Additionally, GANs have been employed as discriminative tools for intrusion detection systems (intrusion detection systems (IDS)) [89, 90], showcasing their versatility in improving security within network systems.

4.3 Reliable Mobile Data Generation

ML-based methods offer flexibility in adapting to dynamic and evolving environments by continuously learning from new data and updating their models accordingly. However, these methods heavily depend on the quality and availability of data. When data is insufficient or inaccurate, it can lead to biased or unreliable results. Acquiring high-quality and representative data presents significant challenges, particularly in domains where data is scarce, expensive, time-consuming to collect, or subject to privacy and security concerns. In such scenarios, generating synthetic data becomes a promising solution to overcome these limitations. Deep generative models have emerged as one of the most exciting sub-fields of deep learning due to their ability to synthesize data by learning the underlying distribution, thus enabling the generation of novel samples [72].
In mobile and IoT networks, data is often generated at the edge, making crowdsourcing a natural and convenient method for data collection. This approach takes advantage of the widespread connectivity and sensing capabilities of devices, creating a collaborative data collection framework. However, crowdsourcing is affected by challenges related to privacy, statistical significance, sampling bias, and the time-consuming nature of data collection and post-processing. This section introduces a novel, comprehensive framework that is independent of specific applications or data types and enables the conditional generation of crowdsourced datasets with location information for mobile and IoT networks. A key feature of the proposed methodology is its ability to assess the uncertainty in newly generated samples, achieved through approximate Bayesian methods. To validate this approach, numerical results are discussed in detail using the illustrative task of minimization of drive test (MDT) data generation.

Motivations and Contributions

Crowdsourced datasets encounter several challenges, which drive the design of the proposed generative framework:

- *Device Heterogeneity*: The quality of crowdsourced data varies due to differences in devices, collection methods, and user behaviors, often requiring extensive post-processing and cleaning. This process can result in data scarcity. Enhancing post-processed data with high-fidelity synthetic data provides a significant advantage.

- *Privacy*: Since data collection in crowdsourcing is conducted by individual users, privacy is a major concern. Synthetic data generation offers a viable solution to address this issue.

- *Statistical Insufficiency*: Motivating users to participate in crowdsourcing measurements is challenging, often leading to regions with insufficient data coverage. Therefore, evaluating the interpolation capabilities of data augmentation methods and assessing uncertainty in data-sparse regions is crucial.

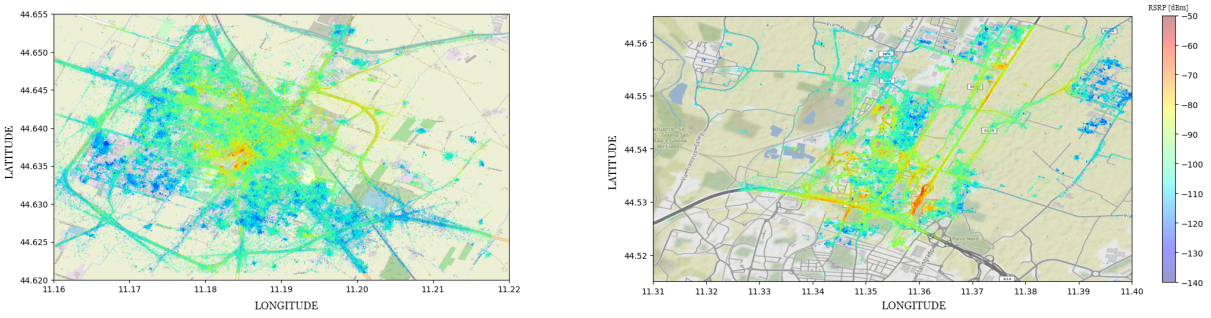
- *Sampling Bias*: Crowdsourced datasets are susceptible to biases related to environmental conditions. For instance, as discussed in Sec. 4.4, RSRQ, a key radio performance indicator, is influenced by network load. Since certain network load conditions are rare, generating synthetic data as a function of these conditions can reduce the need for time-consuming measurement campaigns.

To address these challenges, this section makes the following key contributions:

- We introduce a conditional generative framework for accurately producing synthetic crowdsourced datasets that include location information in mobile and IoT networks. The framework offers two notable features: (1) it enables uncertainty evaluation during the data generation process, which can be broken down into epistemic and aleatoric confidence intervals using approximate Bayesian methods, thus ensuring a *reliable* and *trustworthy* generation process, and (2) it allows for the generation of new samples conditioned on environmental factors, such as average network traffic load.

- The proposed method is validated through the illustrative task of MDT data generation. For the first time, a detailed comparison is provided between the generated data and a large-scale dataset of original MDT measurements collected from an MNO's network infrastructure. This comparison leverages various metrics, with a focus on the algorithm's *robustness*, assessed in terms of *calibration* and interpolation capabilities, as well as the evaluation of uncertainty in data-extrapolated regions. Although the analysis is tailored to MDT data generation under adjustable network and traffic conditions, the proposed methodology is broadly applicable for the generation and augmentation of other crowdsourced datasets with location information.

- Additionally, numerical results on downstream tasks using the generated dataset are provided. These results demonstrate that performance comparable to that achieved with large-scale original MDT datasets can be obtained. The task of fingerprinting-based localization is presented as an illustrative example.



(a) San Giovanni in Persiceto, Italy, with coverage from two tri-sectorial sites (6 eNBs).

(b) Peripheral area north of Bologna, Italy, covered by six tri-sectorial sites (18 eNBs).

Figure 4.1: Reference scenario - Map of geolocated reference datasets.

4.4 System Model

As discussed in the previous section, the proposed framework is introduced using the specific case of MDT data generation. For this, we rely on MDT data collected from an MNO's network infrastructure, covering different scales of deployment. In particular, the data are collected from different urban environments, as shown in Fig. 4.1.
MDT data consists of a rich set of radio features. In this context, the focus is on generating a representative set of radio indicators. Although some indicators were previously mentioned, they are briefly reintroduced here to ensure the section is self-contained:

– **RSRP of the serving cell**: The RSRP is defined as a narrow-band power measurement estimated by the UE based on the channel reference signal (CRS) sent over specific resource elements (REs) in the downlink. These signals are spread across multiple resource blocks, with a pattern determined by the cell's physical cell ID (PCI), and the UE detects these signals within the received OFDM symbols. The RSRP is expressed as the sum of the power carried by individual REs, denoted as $P_{\text{RE},i}$, divided by the number N of sub-carriers carrying CRS across the entire system bandwidth [62]:

$$\text{RSRP} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{14} P_{\text{RE},ik} \ , \tag{4.1}$$

where the sum over k accounts for the number of orthogonal frequency division multiplexing (OFDM) symbols in each time transmission interval (TTI), and $P_{\text{RE},ik}$ is defined as:

$$P_{\text{RE},ik} = \begin{cases} P_{\text{RE},i} & \text{if the } k\text{-th OFDM symbol carries CRS} \\ 0 & \text{otherwise .} \end{cases} \tag{4.2}$$

The RSRP is a critical metric used in various RRM procedures, such as mobility management and power control. As a linear average of independent power samples, the RSRP estimates the median power component observed by the UE. For a detailed derivation of the error lower bound on estimation accuracy, readers can refer to Appendix A.1.

– **RSRP of neighbor cells**: MDT measurements also capture the RSRP from neighboring cells within visibility range. This provides valuable insights into interference patterns and supports network optimization, resource allocation, and mobility management.

– **RSRQ**: The RSRQ is inversely proportional to the received signal strength indication (RSSI), a wide-band measure capturing the power from both serving and non-serving cells:

$$\text{RSSI} = \sum_{i=1}^{M} \sum_{j=1}^{12} \sum_{k=1}^{14} P_{\text{RE},ijk} \ . \tag{4.3}$$

In (4.3), M represents the number of PRBs over the system bandwidth, and the sum over j covers all OFDM subcarriers in a PRB. The RSRQ is then expressed as the product of the number M of PRBs and RSRP, divided by RSSI:

$$\text{RSRQ} = \frac{M \cdot \text{RSRP}}{\text{RSSI}} \ . \tag{4.4}$$

The RSRQ provides key statistics about network usage, with the average load ρ being inversely correlated with RSRQ.

– **User association**: Each MDT measurement is associated with a serving base station. This is essential for observing traffic distribution and mobility load balancing. For accurate analysis, each newly generated sample must be associated with a serving base station.

The proposed approach can be extended to different feature sets. The chosen indicators represent a mix of KPIs that either depend (e.g., RSRQ) or do not depend (e.g., RSRP) on external factors like network load ρ. These factors may introduce sampling bias in the original dataset. The mobile data generation process is handled through sub-problem decomposition. The generation and clustering of space-time-dependent traffic samples are treated independently from the probabilistic regression of radio features. This approach captures geo-related dependencies such as radio environment characteristics, line-of-sight (LoS)/non-line-of-sight (NLoS) conditions, and clutter during the training of regression models. Fig. 4.2 and Fig. 4.3 illustrate the system architecture for both training and inference phases.

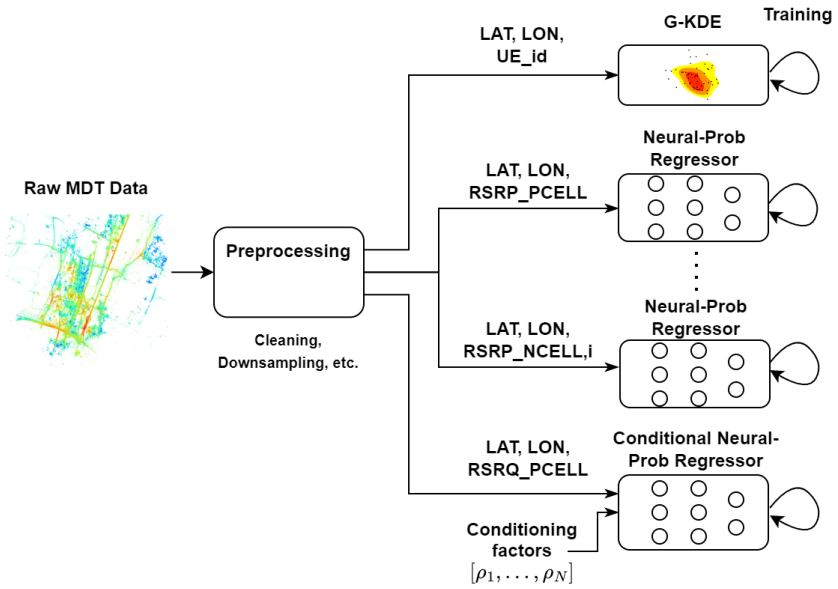


Figure 4.2: System model - Training architecture.

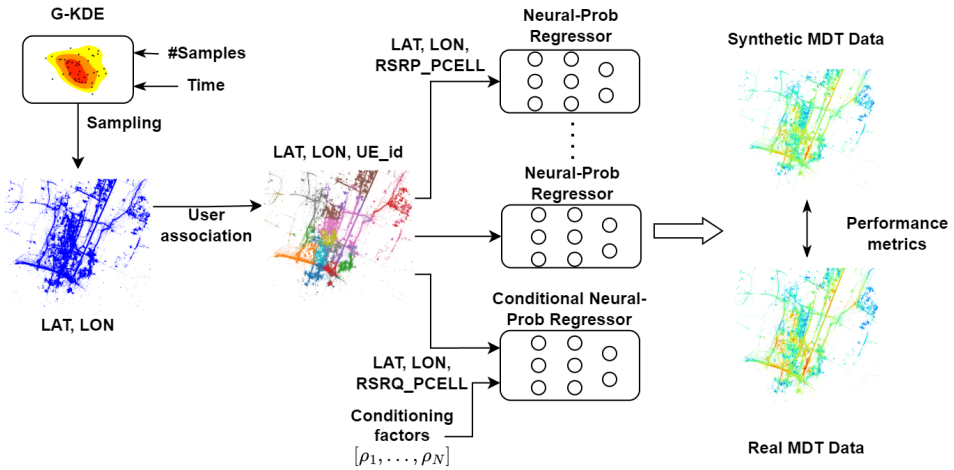


Figure 4.3: System model - Inference architecture.

Training

The generation of space-time-dependent user samples and user association is framed as a trivariate density estimation problem over time, latitude, and longitude. User association plays an important role in generating synthetic MDT data, though this may not be necessary for other datasets or applications where preserving the serving base station information is less critical. Formally, the density estimation problem is expressed as a maximum likelihood problem (or

equivalently, a minimum negative log-likelihood problem):

$$\arg\min_{\theta} \ \mathbb{E}_{x\sim P(x)}\left[-\log f(x|\theta)\right], \tag{4.5}$$

where $x = (t, \text{lat}, \text{lon})$ is the sampled data, $P(X)$ is its distribution, θ represents the model parameters, and f is the likelihood of the data x given the model parameters θ. The goal is to minimize the expected negative log-likelihood.
In practice, this becomes the minimization of the loss function:

$$\arg\min_{\theta} \ \mathcal{L}(\theta) = \sum_{i=1}^{m} -\log f(x_i \mid \theta), \tag{4.6}$$

where $\{x_i\}_{i=1}^{m} = \mathcal{D}$ is the training set. A held-out validation set is typically used to fine-tune model parameters.
For the probabilistic regression of radio features, the focus is on assessing uncertainty in predictions. A Bayesian approach is used for probabilistic regression, with models trained via stochastic variational inference (SVI) by minimizing the variational free energy:

$$\arg\min_{\lambda} \ \text{KL}[Q_\lambda(\theta)\|P(\theta)] + \mathbb{E}_{\theta\sim Q_\lambda(\theta)}\left[\mathcal{L}(\theta)\right], \tag{4.7}$$

where $\mathcal{L}(\theta)$ denotes any suitable regression loss function, and the expectation is computed using Monte Carlo methods. Additional details on approximate Bayesian methods are provided in Sec. 2.2 and Sec. 4.5.
Each problem is addressed using independent learners. After pre-processing, the original MDT dataset is fed into the blocks shown in Fig. 4.2. External conditioning factors like network load ρ are fed into regressors for conditional regression tasks (e.g., RSRQ), as detailed in Sec. 4.5.

Inference

After training, the inference phase proceeds through a pipeline (shown in Fig. 4.3) involving: (i) sampling generated users in the time-space domain, (ii) user association, and (iii) probabilistic (conditional) regression of radio features. A synthetic MDT sample includes the following artificial features:

- sample_latitude
- sample_longitude
- sample_serving_cell_ID
- sample_primary_RSRP

- sample_neighbour_RSRP_1, . . . , N
- sample_neighbour_RSRQ.

4.5 Algorithms and performance metrics

This section presents the theoretical foundations, design principles, and training processes involved in the building blocks of the data generation framework.

Sample Generation and User Association via Gaussian Kernel Density Estimation

Data augmentation for tabular datasets can be tackled through various approaches, many of which are available in ready-to-use libraries [91]. Among the state-of-the-art techniques are Bayesian networks (BNs) [92], GANs, and variational autoencoders (VAEs) [93]. However, the spatial distribution of MDT data introduces specific challenges for these models due to its inherent complexity and irregularity, which are influenced by the topography of the targeted area. To address this issue, a non-parametric density estimation method known as kernel density estimation (KDE) is employed. KDE provides a smooth estimation of the probability density function (pdf) by leveraging all sample points' locations, allowing for the detection of multimodal patterns [94].

Given a sequence of samples $\mathcal{D} = \{x_i\}_{i=1}^{m}$, where each sample $x_i = \{\text{lat}_i, \text{lon}_i\}$ comes from a distribution $P(x)$, the estimated density $f(x, \mathcal{D}, h) : \mathbb{R}^{2\times n} \rightarrow \mathbb{R}^{+}$, which approximates the true distribution $f^{*}(x)$, is computed as:

$$f(x, \mathcal{D}, h) = \frac{1}{n} \sum_{i=1}^{n} K(x - \hat{x}_i, h) \tag{4.8}$$

where

$$K(x - x_i, h) = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{(x - x_i)^2}{h^2}\right) \tag{4.9}$$

is the Gaussian kernel function, and x is the point at which the pdf is estimated. The model parameters are derived from the n training samples $\mathcal{D}$, and h is the smoothing parameter, also referred to as bandwidth.

The fundamental assumption of KDE is that a higher sample density in a particular region indicates a higher probability of observing new samples in that vicinity.

KDE is an unsupervised learning technique, and its bandwidth parameter h is optimized using empirical risk minimization (ERM) on a validation set $\mathcal{D}^{val} = \{x_i\}_{i=1}^{n_{val}}$ with n_{val} samples. Specifically, h is chosen to minimize the negative log-likelihood (4.5) of Gaussian kernel density estimation (G-KDE) on $\mathcal{D}^{val}$:

$$\arg\min_{h} -\sum_{i=1}^{n_{val}} \log f(x_i, \mathcal{D}, h) \ . \tag{4.10}$$

The distribution of users across an area also exhibits temporal dependency. However, within sufficiently short time intervals, $f(x, \mathcal{D}, h)$ can be assumed to be time-invariant. Therefore, N represents the required number of time windows to capture independent, time-invariant user probability distributions. The outcome is a tri-variate density distribution, continuous in space and having N possible discrete values in time.

For each time window N, K distinct KDE models are trained using samples from each of the K primary cells (PCELLs) in the area. This step is critical for correctly associating newly generated samples $(x'_{\text{LAT}},\ x'_{\text{LON}})$ with their serving PCELL. Let f_j^O denote the overall density estimation for the time window $j \in \{1, ..., N\}$, and $f_j^{C_i}$ the density estimation trained on samples from PCELL$_i$, with $i \in \{1, ..., K\}$. The association process consists of assigning each new sample $(x'_{\text{LAT}},\ x'_{\text{LON}}) \sim f_j^O$ to the PCELL that corresponds to the distribution $f_j^{C_i}$ with the highest likelihood (4.11):

$$\text{PCELL}' = \arg\max_{i} \left\{ \log f_j^{C_i} \left(x'_{\text{LAT}},\ x'_{\text{LON}}\right), \text{where } \left(x'_{\text{LAT}},\ x'_{\text{LON}}\right) \sim f_j^O(\hat{\mathbf{x}}_{\mathbf{ij}}, h) \right\}, \tag{4.11}$$

where $\hat{\mathbf{x}}_{\mathbf{ij}}$ represents the original MDT samples associated with PCELL$_i$ for time window j. The complete generation process of new samples $(x'_{\text{LAT}},\ x'_{\text{LON}},\ \text{PCELL}')$ is outlined in Algorithm 2, and an illustrative example is shown in Fig. 4.4.

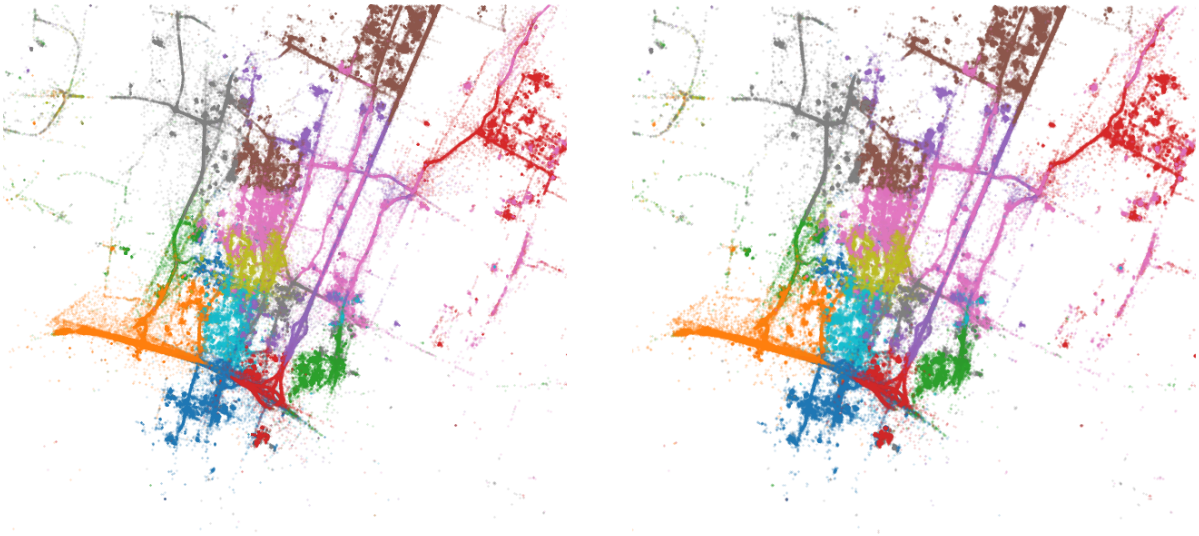
Algorithm 2 G-KDE Sample Generation

Require:

$j \in \{1, .., N\}$, the selected time window
$\mathcal{S}$, the total set of samples to be generated
for $s \in \mathcal{S}$ **do**
    Initialize $(x'_{\text{LAT}},\ x'_{\text{LON}})_s \sim f_j^O$
    for each $i \in \{1, ..., K\}$ **do**
        $\mathcal{L}(i) = \log\left(f_j^{C_i}(x'_{\text{LAT}},\ x'_{\text{LON}})_s\right)$
    end for
    $\text{PCELL}'_s = \arg\max_i\{\mathcal{L}(1) \dots \mathcal{L}(K)\}$
    Assign $(x'_{\text{LAT}},\ x'_{\text{LON}})_s$ to PCELL'_s
end for

RSRP: Bayesian Neural-Probabilistic Regression

To address feature regression, a Bayesian neural-probabilistic regression is employed. This regression approach combines neural networks with variational inference (VI) (Sec. 2.2) to



(a) Ground-Truth: Each MDT sample is associated with its serving PCELL, represented by different colors.

(b) Synthetic MDT samples after G-KDE sampling and the user association process described in Algorithm 2.

Figure 4.4: Illustrative results of MDT sample generation and user association.

perform a regression task. The final layer of the neural network is modeled as a parameterized probability distribution $P(y \mid x, \theta)$. As previously discussed in Sec. 2.2, the model is trained by minimizing the variational free energy cost function:

$$KL[Q_\lambda(\theta) \,||\, P(\theta)] - \mathbb{E}_{\theta \sim Q_\lambda, y \sim P(y|x)}[\log P(y \mid x, \theta)] \,, \tag{4.12}$$

where the first term represents the Kullback-Leibler divergence between the variational posterior and prior distributions, while the second term represents the negative log-likelihood. The expectation over y is handled via ERM.

Handling the expectation over θ and the first loss term requires defining a prior distribution for the model weights and specifying a parametric assumption for the output distribution. A common choice for the prior is an isotropic Gaussian distribution with covariance matrix $\mathbb{K}_{\bar{\theta}} = \sigma^2 I$. However, special attention must be paid to the parametric assumption for the final distribution. If this assumption is not well-aligned with the true distribution of the target variable, model miss-specification may occur, which can degrade the model's calibration performance. For the task of RSRP regression, the final layer is modeled as a Gaussian distribution. For a detailed analytical derivation, see Appendix A.1. This assumption is further validated by numerical results in Sec. 4.6, showing that the proposed model is calibrated by design, avoiding issues with model miss-specification.

A concise formulation of the neural-probabilistic model can be expressed as:

$$\hat{y} \sim \mathcal{N}\left(\mu(x, Q_\lambda(\theta)), \sigma(x, Q_\lambda(\theta))\right) \tag{4.13}$$

In this expression, a predicted sample $\hat{y}$ is drawn from a Normal distribution, where the mean μ and standard deviation σ are functions of the variational distribution $Q_\lambda(\theta)$ and the input x. This probabilistic framework enables the model to capture both aleatoric and epistemic uncertainties simultaneously. However, for interpretability or practical purposes, it is often useful to decompose these two types of uncertainty.

Given an ensemble of M probabilistic models, $\{P(y \mid x, \theta_i \sim Q_\lambda(\theta))\}_{i=1}^{M}$, one effective method for decomposing uncertainty is the law of total variance [95]:

$$\underbrace{\mathbb{V}_{P(y|x,\mathcal{D})}(y)}_{\text{total uncertainty } \varepsilon} = \underbrace{\mathbb{V}_{P(\theta|\mathcal{D})}(\mathbb{E}_{P(y|x,\theta)}[y])}_{\text{epistemic uncertainty } \varepsilon_{ep}} + \underbrace{\mathbb{E}_{P(\theta|\mathcal{D})}[\mathbb{V}_{P(y|x,\theta)}(y)]}_{\text{aleatoric uncertainty } \varepsilon_{al}}, \tag{4.14}$$

where $\mathbb{V}(y)$ is the variance of y and $\mathbb{E}(y)$ its expectation. For the Gaussian parameterization used in (4.13), Eq. (4.14) can be approximated using Monte Carlo sampling:

$$\mathbb{V}_{P(y|x,\mathcal{D})}(y) \approx \frac{1}{M}\sum_{i=1}^{M}\left[\mu_M - \mu_i\right]^2 + \frac{1}{M}\sum_{i=1}^{M}\sigma_i^2 \,. \tag{4.15}$$

In this approximation, μ_i and σ_i are the mean and variance of the i-th probabilistic regression model from the Bayesian ensemble and $\mu_M := \frac{1}{M}\sum_{i=1}^{M}\mu_i$. This decomposition provides a practical way to separate epistemic uncertainty from the inherent noise in the data. Moreover, it can be used to assess the *trustworthiness* of model predictions in challenging scenarios, as demonstrated in Sec. 4.6.

An additional important design consideration for the proposed model is the choice of input features. Here, the inputs consist of device geolocation data—latitude (LAT), longitude (LON). While these features are simple, they implicitly capture location-dependent factors that affect RSRP, such as propagation environments, line-of-sight (LoS)/non-line-of-sight (NLoS) conditions, surrounding clutter, and building materials. This allows the model to infer these dependencies without needing additional contextual environmental information. By contrast, RSRQ is influenced by factors such as the average cell load ρ, which are not strictly tied to location. These additional factors must be provided explicitly as inputs for conditional regression, as discussed in the following section.

The overall model architecture is shown in Fig. 4.5.

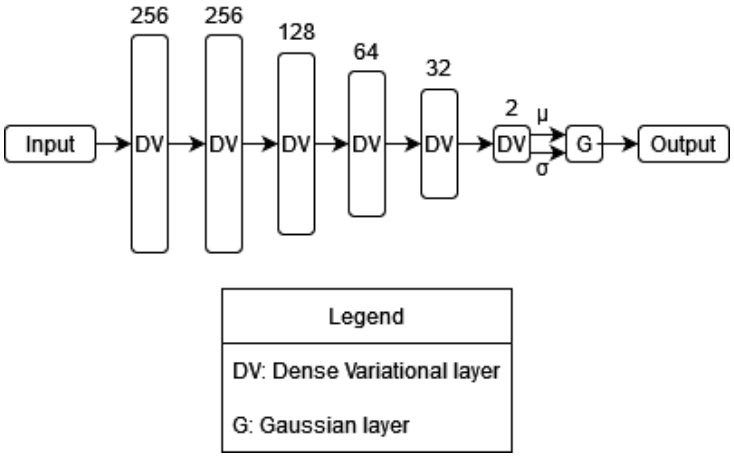


Figure 4.5: Bayesian neural-probabilistic model architecture

RSRQ: Conditional Bayesian Neural-Probabilistic Regression

In machine learning, dataset imbalance is a common issue that often leads to discrimination against underrepresented classes [96]. When performing probabilistic regression of KPI features in a crowdsourcing environment, this challenge is further complicated by sampling bias, which must be carefully addressed. This issue is particularly significant for KPI features that depend on non-location-specific factors, such as RSRQ. Since extreme traffic conditions (either very high or very low) are rare, RSRQ samples collected under these conditions are typically underrepresented.

To mitigate this imbalance, one effective solution is to apply inverse probability weighting (IPW). When implemented correctly, IPW can enhance the efficiency and reduce the bias of unweighted estimators. Technically, IPW introduces a per-sample weighted cost function, where each sample weight α_i is proportional to the inverse of the probability of observing that sample in the training set. For instance, when considering the log-likelihood cost function, the modified cost function with IPW is given by:

$$\mathcal{L}(\theta) = \log \prod_{i=1}^{|\mathcal{D}|} \alpha_i P(y_i \mid x_i, \theta) = \sum_{i=1}^{|\mathcal{D}|} \log \alpha_i P(y_i \mid x_i, \theta) \,, \tag{4.16}$$

where the weight α_i is defined as:

$$\alpha_i \propto \frac{1}{P(x_i \mid \mathcal{D})} \,. \tag{4.17}$$

Here, $P(x_i \mid \mathcal{D})$ is the probability of observing the input value x_i, estimated based on the training dataset $\mathcal{D}$. This probability is approximated using a histogram-based approach, where the input values x are first discretized into bins, and categorical probabilities are then computed.

The choice of how to discretize the load space into bins is a design consideration.
By employing IPW, the objective function becomes biased towards underrepresented samples, improving fairness in the regression process and addressing the issue of skewed data distributions.
When weights become excessively large due to low probability estimates, they can significantly increase the variance of the regression model. This may lead to unstable learning dynamics and potential overfitting to noise in regions with sparse data. To mitigate this issue, smoothing techniques or regularization strategies can be employed. For instance, one can apply kernel density estimation or incorporate Bayesian priors to prevent the estimated probabilities from being overly influenced by the scarcity of data for certain regions. Additionally, techniques like weight clipping (i.e., setting an upper limit on α_i) can be useful to prevent any single sample from dominating the learning process.

Performance Metrics

This section introduces the metrics used to evaluate the performance of the proposed generative algorithmic framework. The focus is on assessing the effectiveness in probabilistic regression, as well as in the downstream task of fingerprinting-based localization, which is performed using the synthetic dataset.

Probabilistic regression

- **MAE** and **RMSE**: Two straightforward metrics for evaluating the effectiveness of the neural-probabilistic regression approaches are MAE and root mean squared error (RMSE), defined as:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} (y_i - \hat{y}_i)^2} \,, \tag{4.18}$$

$$\text{MAE} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} |y_i - \hat{y}_i| \,, \tag{4.19}$$

  where $\hat{y}_i$ is the estimated value for the i-th sample and y_i is the ground truth. However, these metrics alone do not capture the probabilistic aspect of the model.

- **Calibration**: Following [97], the quality of probabilistic regression is also evaluated in terms of calibration plots and calibration error. Calibration plots show the true frequency of points within each confidence interval versus the predicted frequency for that interval, computed as:

$$\tilde{P}_j = \frac{|\{y_i \mid F_i(y_i) \leq P_j,\ i \in \mathcal{D}_{test}\}|}{|\mathcal{D}_{test}|} \,, \tag{4.20}$$

where P_j is the true frequency for a given quantile $j \in \{0, \dots, 1\}$, $\tilde{P}_j$ is the empirical frequency for that quantile, and $F_i(y_i)$ is the cumulative distribution function (CDF) for the probabilistic output given input x_i. For a Gaussian-parametrized output (4.13), $F_i(y_i)$ can be computed as:

$$\begin{aligned} F_i(y_i) &= P\left(\mathcal{N}\left(\mu(x_i, Q_\lambda(\theta)), \sigma(x_i, Q_\lambda(\theta))\right) \leq y_i\right) = \\ &= \frac{1}{\sigma_i\sqrt{2\pi}} \int_{-\infty}^{y_i} \exp \frac{(x-\mu_i)^2}{\sigma_i^2}\, dx. \end{aligned} \tag{4.21}$$

A calibration error (CE) score can then be computed to quantify the model's calibration ability:

$$\text{CE} = \sum_j w_j |\tilde{P}_j - P_j| \,, \tag{4.22}$$

where $w_j \propto |\{y_i \mid F_i(y_i) \leq P_j,\ i \in \mathcal{D}_{test}\}|$ ensures that quantiles with fewer samples are given less importance.

– **Sharpness**: As suggested by [97], sharpness evaluates how tightly the probabilistic model bounds its predictions, measured by the predictive standard deviation $\sigma(x_i, Q_\lambda(\theta))$, which accounts for both epistemic and aleatoric uncertainties:

$$S = \sigma(x^{\text{test}}, Q_\lambda(\theta)) \,, \tag{4.23}$$

where x^{test} is the vector of test data input values, and S is a vector with dimension $\mathbb{R}^{1\times m^{\text{test}}}$, where $m^{\text{test}} = |x^{test}|$. To derive scalar metrics from (4.23), we compute the average sharpness over the test set, $\mathbb{E}[S]$, and the sharpness standard deviation $\sigma[S]$:

$$\mathbb{E}[S] = \frac{1}{m^{\text{test}}} \sum_{i=1}^{m^{\text{test}}} \sigma(x_i, Q_\lambda(\theta)) \,, \tag{4.24}$$

$$\sigma[S] = \sqrt{\frac{1}{m^{\text{test}}} \sum_{i=1}^{m^{\text{test}}} \left(\sigma(x_i, Q_\lambda(\theta)) - \mathbb{E}\left[\sigma(x_i, Q_\lambda(\theta))\right]\right)^2} \,. \tag{4.25}$$

A model with narrower sharpness produces more informative predictions, assuming equivalent calibration.

– **Average epistemic uncertainty** ε_{ep}: Finally, the average epistemic uncertainty is a key metric derived from (4.13) and (4.14). Since aleatoric uncertainty ε_{al} is irreducible, a model's ability to minimize epistemic uncertainty directly reflects its reliability. The average epistemic uncertainty over a set of L locations $X_A = \{x_1, \dots, x_L\}$ in an area

A can be computed as:

$$\mathbb{E}_A[\varepsilon_{ep}] = \mathbb{E}_A\left[\mathbb{V}_{P(\theta|\mathcal{D})}\left(\mathbb{E}_{P(y|x,\theta)}[y]\right)\right] = \frac{1}{LM}\sum_{j=1}^{L}\sum_{i=1}^{M}\left[\mu_{M,j} - \mu_{i,j}\right]^2 , \tag{4.26}$$

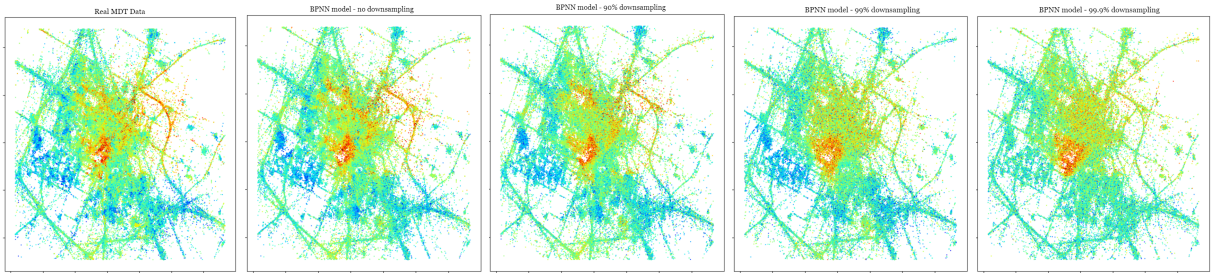
where $\mu_{i,j}$ refers to $\mu(x_{i,j}, Q_\lambda(\theta))$ for simplicity, $\mu_{M,j} := \frac{1}{M}\sum_{i=1}^{M}\mu_{i,j}$, and M is the number of Monte Carlo experiments performed for each x_j. This metric helps assess the model's ability to express uncertainty in extrapolation scenarios and defines a threshold for distinguishing between "reliable" and "unreliable" predictions.

MDT-based fingerprinting

In addition to evaluating probabilistic regression, the quality of the synthetic data is assessed through a downstream task on fingerprinting-based localization using the synthetic dataset. Performance is measured by comparing the RMSE (4.18) achieved on the original dataset with that obtained on the synthetic dataset, using identical models for both. In this task, the model provides a point estimate of the ground truth variable y, which represents the true position (latitude, longitude).

4.6 Numerical Results

This section presents the results of the key metrics discussed earlier and compares the generated data to a large-scale original dataset of MDT data from a MNO's network infrastructure. To comprehensively evaluate each model characteristic, the section is organized into the following sub-sections: Interpolation, Extrapolation, Conditional Generation, and Downstream Tasks.



(a) Ground-Truth (b) Original training set (c) 10% training set size (d) 1% training set size (e) 0.1% training set size

Figure 4.6: Visual comparison of ground-truth values and RSRP predictions from a Bayesian neural-probabilistic model trained on downsampled training sets.

Metric	Full training set	Downsampling 90%	Downsampling 99%	Downsampling 99.9%
MAE [dB]	5.42	5.92	5.93	6.64
CE	2.09e-2	2.27e-2	2.67e-2	4.98e-2
S ($\mathbb{E}[S]$ [dB], $\sigma[S]$ [dB])	(7.12, 1.55)	(7.39, 1.43)	(8.23, 1.25)	(8.5, 0.95)

Table 4.1: Numerical comparison of ground-truth values and RSRP predictions under increasing downsampling.

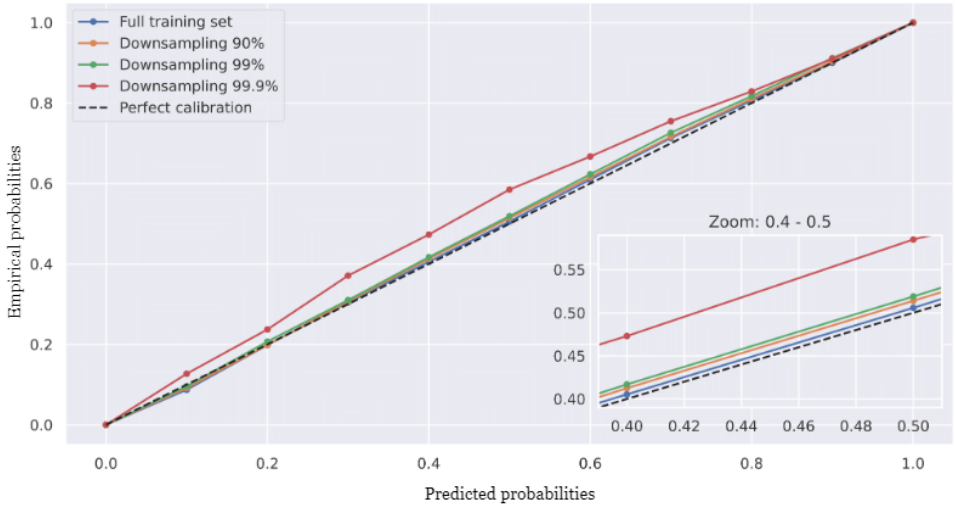


Figure 4.7: Calibration plot for BPNNs under increasing downsampling, showing Gaussian distribution assumptions are confirmed.

Interpolation

This subsection evaluates the interpolation capabilities of the proposed Bayesian neural probabilistic model. Using a 65/35 train-test split of the dataset, the model is trained on the urban scenario depicted in Fig. 4.1a. The performance metrics—MAE, Calibration Error (CE), and Sharpness—are evaluated as a function of decreasing training set size (see Tab. 4.1). Multiple independent training runs are conducted with different downsampling rates applied to the training data. Performance is assessed on a held-out test set that includes the original sample locations and RSRP values.

Fig. 4.6 and Table 4.1 show that as the training set size decreases, the predictive performance drops slightly. For instance, with 99.9% downsampling, the MAE only increases by 1.22 [dB] compared to the full dataset. Fig. 4.7 illustrates the model's calibration as downsampling increases.

As shown in Fig. 4.7, the model becomes more under-confident as downsampling increases, aligning with the desired cautious behavior in data-scarce scenarios. This supports two key conclusions: the model is *robust* to substantial downsampling and demonstrates reliable uncertainty-aware predictions when data is sparse. Additionally, the model's out-of-the-box calibration, as

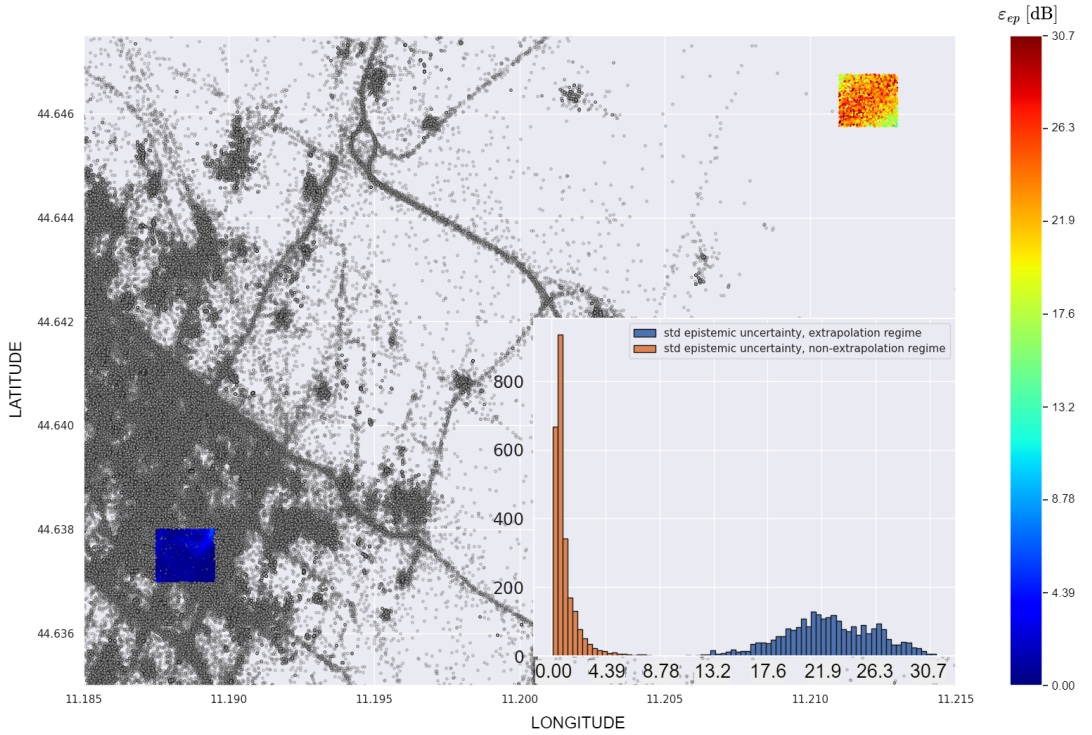


Figure 4.8: Distribution of epistemic uncertainty in extrapolation vs non-extrapolation regions.

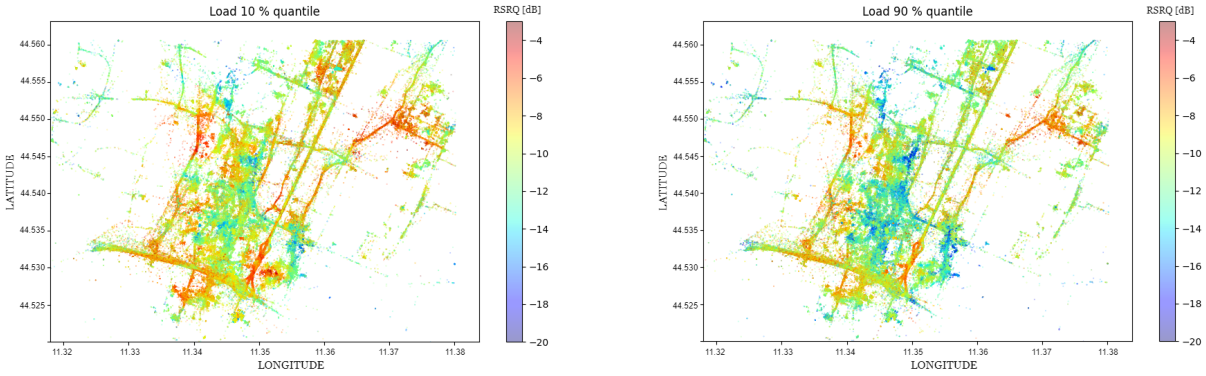
seen in Fig. 4.7, empirically supports the assumption of Gaussian parameterization of the RSRP distribution.

Extrapolation

In addition to its cautious behavior when confronted with sparse data, the model effectively conveys uncertainty in extrapolation regions. A neural probabilistic model trained on the urban scenario in Fig. 4.1a is used for inference in two areas: one with dense data and the other sparsely sampled (see Fig. 4.8). The results show higher epistemic uncertainty in extrapolation areas, where $\mathbb{E}_{A2}[\varepsilon_{ep}] = 21.9$ [dB] compared to 0.87 [dB] in non-extrapolation regions.
This distinction is further highlighted in Fig. 4.8, where the epistemic uncertainty distributions for the two regions are shown. This finding suggests the use of epistemic uncertainty thresholds to classify new data points as "reliable" or "unreliable," facilitating strategic planning for new measurements in crowdsourced settings. This framework is aligned with the principles of active learning [98], where data collection incurs costs such as drive tests or communication.

Conditional Generation

A notable feature of the proposed generative framework, as discussed in Sec. 4.5, is its ability to generate probabilistic outputs conditioned on factors that are independent of sample loca-



(a) Low ρ (10% quantile)

(b) High ρ (90% quantile)

Figure 4.9: Conditional probabilistic regression of RSRQ based on different configurations of ρ.

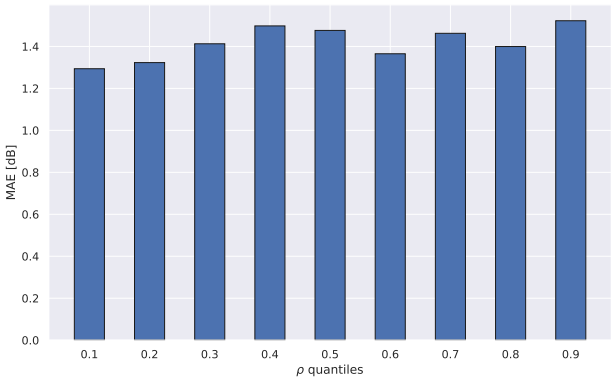


Figure 4.10: MAE on RSRQ for the non-extrapolation regime.

tions and have varying probabilities of occurrence. This capability addresses the challenge of sampling bias by mitigating its impact. Specifically, we evaluate the framework's neural probabilistic regressor in generating diverse RSRQ values, conditioned on the average cell load. The focus is on assessing the MAE within the non-extrapolation regime. Fig. 4.10 shows the MAE as a function of quantiles of the average cell load in the training set, where cell-level variations may occur. The use of IPW proves beneficial in ensuring fairness, especially in underrepresented quantiles such as the 10% and 90% quantiles.
Additionally, Fig. 4.9 visually illustrates the influence of conditioning on ρ on predicted RSRQ values, where higher loads correspond to lower predicted values.

Downstream Task: Localization via Fingerprinting

ML-based fingerprinting [99] is a technique that uses ML algorithms to determine the location of a UE by analyzing radio frequency (RF) signals. It involves two phases: during the *offline*

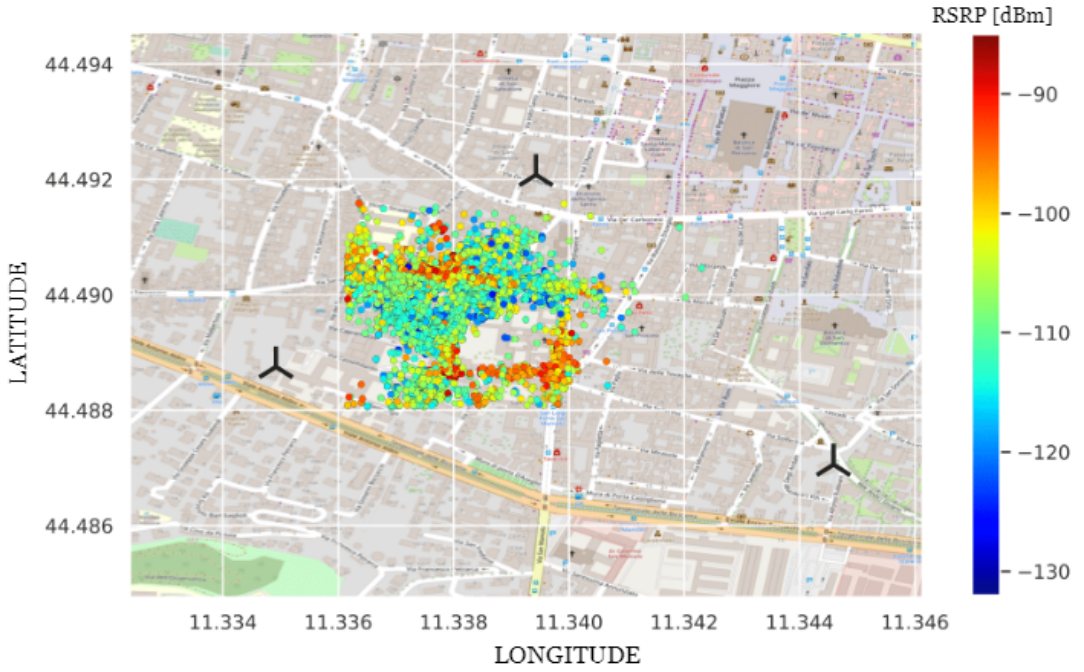


Figure 4.11: Reference scenario for fingerprinting-based localization experiments. MDT data with RSRP samples from the three cells shown, collected in the city center of Bologna, Italy.

phase, an ML algorithm is trained on a database of RF fingerprints—unique representations of signal characteristics at known locations. In the *online* phase, the trained model infers location based on newly observed fingerprints.

Here, we examine ML-based fingerprinting localization using datasets that include both original MDT fingerprints and synthetically generated ones. Synthetic fingerprints are created by generating new samples in the space-time domain via G-KDE, followed by probabilistic regression of their features ($\text{RSRP}_{1,\dots,N}$), as depicted in Fig. 4.3. For the experiments, we focus on a dense urban area in the city center of Bologna, Italy, as illustrated in Fig. 4.11. The dataset contains 11,000 samples, split 80/20 into training and test sets.

RF fingerprints, including the RSRP measurements from the three e-NodeBs (eNBs) in Fig. 4.11, are defined as:

$$\text{RF}_i = \{\text{RSRP}_{\text{i, A}}, \text{RSRP}_{\text{i, B}}, \text{RSRP}_{\text{i, C}}; \{\text{LAT}_i, \text{LON}_i\}\} \tag{4.27}$$

A Random Forest is used as ML algorithm for regression, trained on both the original training set (approximately 8.5K samples) and a synthetic training set generated from the original data, each containing the same number of samples. Fig. 4.12 presents the results from both ML models on the original held-out test set.

The Random Forest regressor trained on both original and synthetic MDT fingerprints yielded RMSE values of 72.56 m and 77.54 m, respectively. These results support the generative framework's effectiveness, as localization based on synthetic samples shows comparable performance

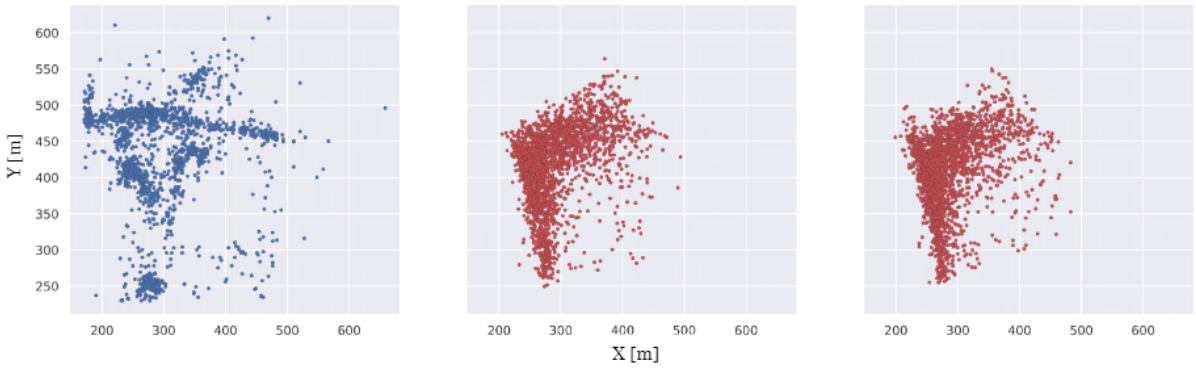


Figure 4.12: From left to right: Original test set, predicted positions based on original MDT fingerprints (RMSE = 72.56 m), predicted positions based on synthetic MDT fingerprints (RMSE = 77.54 m).

to models trained on original data.

Further experiments simulate received power samples, replacing real RSRP measurements. Power samples p_i are generated as per Eq. (4.28), with fixed transmit power $p_0 = 10$ [dB], an exponent $\beta = \{2, 4\}$, and varying shadowing standard deviation σ_S [dB]:

$$p_{i,A} = p_0 - \left(\frac{4\pi d_A}{\lambda}\right)^{\beta} + n \sim \mathcal{N}(0, \sigma_S^2) \; . \tag{4.28}$$

The corresponding fingerprints are given by:

$$\text{RF}_i = \{p_{i,A}, p_{i,B}, p_{i,C}; \{\text{LAT}_i, \text{LON}_i\}\} \; . \tag{4.29}$$

The primary goal is to evaluate the generative framework's performance under different levels of variability (σ_S) in the target variable. Fig. 4.13 shows error curves as a function of σ. The Random Forest regressor trained on synthetic data performs similarly to the one trained on original simulated fingerprints, with only minimal degradation (< 1 [m] on average). The slightly larger discrepancy observed with real RSRP samples (< 5 [m]) is likely due to location-specific noise σ in the crowdsourced data, as opposed to the constant σ_S used in the simulated scenario. These results further validate the framework's versatility across both real and simulated geo-located datasets.

Conclusion

The findings presented in this section highlight the suitability of the proposed framework for the synthetic generation and augmentation of real-world crowdsourcing datasets. The framework demonstrates strong interpolation capabilities, with minimal performance degradation (only 1.22 [dB]) when trained on a downsampled dataset (from 1M to 1K samples). Furthermore,

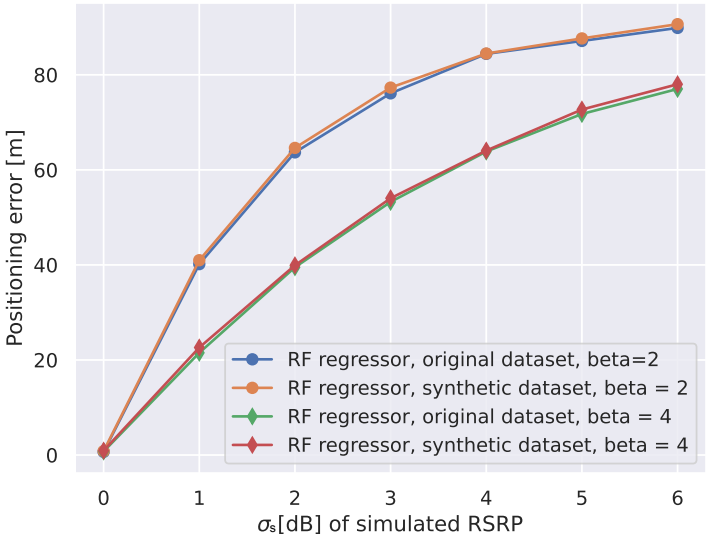


Figure 4.13: Fingerprinting results: RF regressor trained on original vs synthetic fingerprints as a function of σ_S and β.

the model shows increased epistemic uncertainty in areas of extrapolation, enhancing its trustworthiness and suitability for planning new measurement campaigns.
The model also excels in conditional regression, accurately predicting rare network conditions such as high or low average loads. Its robustness to misspecification is supported through both analytical and empirical evidence. Additionally, synthetic samples generated by the model retain comparable performance on downstream tasks like fingerprinting-based localization.
Future work could explore applying this generative framework in an online active learning context, where distributed agents collect data. In such scenarios, the framework's ability to quantify uncertainty in newly generated samples could help balance the tradeoff between generating synthetic data (which incurs no cost) and collecting new data, which incurs communication costs.

Chapter 5

Distributed Learning for Radio Resource Management

In this chapter, we explore one of the core themes of this thesis: how "AI-native communication systems" are poised to revolutionize next-generation autonomous networks. At the heart of this transformation lies the convergence of distributed learning and wireless communications, which represents a fundamental paradigm shift. In conventional wireless networks, the primary focus is on data transmission and reception, with information flowing in a unidirectional manner. However, 6G autonomous networks are envisioned to support the seamless exchange of data, knowledge, and decision-making capabilities ubiquitously. Wireless nodes will evolve into intelligent entities that not only transmit data but also communicate their intents and learn from the data they receive. As a result, wireless networks must become more data-efficient and adaptive, while machine learning principles must also evolve to leverage the unique opportunities and challenges inherent in wireless environments.

In the context of distributed learning, the design focus for 6G networks has shifted from maximizing data rates to accelerating the training of ML models using distributed data sources [100]. Here, performance is measured by how communication influences a collaborative goal (e.g., the training of an ML model), rather than by the accuracy of symbol transmission from one point to another. This aligns with the *effectiveness problem*, first articulated by Shannon and Weaver in *"A Mathematical Theory of Communication"*. Traditional wireless networks were not designed with *effective*, or *goal-oriented*, communication in mind. As such, the design of communication systems that reliably transmit signals has often been treated separately from the "language" necessary for achieving coordination and cooperation among agents [20]. The future emphasis on goal-oriented communications underscores the need for novel algorithms and techniques that integrate communication and learning seamlessly.

A key area of focus in this chapter is multi-agent reinforcement learning (MARL), which highlights the integration of communication and learning to achieve specific goals. MARL involves

a set of distributed devices that work toward a common optimization objective within the framework of a Markov game. One notable finding in this field is that the challenges traditionally associated with MARL can be mitigated by enabling communication among the agents involved [23, 101].
In summary, the integration of distributed learning and wireless networks is set to reshape both fields. This convergence, driven by the principles of communication and learning working together, promises more efficient and adaptable wireless networks, along with the development of ML algorithms that can thrive in dynamic and resource-constrained environments. The following chapters will delve deeper into the theoretical underpinnings, practical implementations, and future possibilities of this emerging paradigm. A central theme throughout the chapter is the strategic use of graph structures as a key tool for efficiently delivering distributed learning over wireless networks, including scalable network optimization and radio resource management via graph-based multi-agent reinforcement learning (Sec. 5.3).

5.1 Leveraging Graph Structures for Distributed Learning

All neural network models inherently introduce a form of inductive bias. This bias is embedded through various design choices, such as activation functions, the hierarchical organization in multi layer perceptrons (MLPs), or the filter sizes in CNNs. However, when the problem domain involves a network with critical structural or topological significance, a relational inductive bias becomes more relevant.
Conventional neural network models often fall short in addressing the challenges posed by wireless networks because they fail to capture the structured relational biases intrinsic to these problems. In scenarios involving multiple distributed nodes within a wireless network—such as resource allocation, power control, scheduling, handovers, or access point (AP) selection—classical DL architectures prove inadequate. These models lack the expressiveness needed to handle the relational dynamics between the nodes.
A promising solution to these challenges, where the network's topology and the mutual relations between distributed wireless nodes are critical, is the adoption of model architectures that are specifically designed to reflect the unique topology and physical dynamics of wireless networks. Since the topology of a wireless network can naturally be represented as a graph, it becomes intuitive to leverage graph-based architectures, such as GNNs, for these tasks. Graphs provide a flexible representation for modeling arbitrary (pairwise) relational structures, and graph-based computations offer a robust relational inductive bias that exceeds what is achievable with conventional convolutional or recurrent layers [102]. For example, the message-passing mechanism, central to GNNs (discussed in Sec. 2.3), propagates information consistently across nodes in a graph. This leads to a modular flow of information that can be effectively applied to graph-

ical models of varying sizes and topologies [102].

5.2 Multi-Agent Systems

A multi-agent system involves multiple autonomous agents interacting within a shared environment to achieve individual or collective goals. In complex systems, such as wireless networks, attempting to program intelligent behavior through traditional methods is extremely difficult, if not impossible. Therefore, agents must adapt and learn autonomously over time. To this end, RL, as discussed in Sec. 2.1, offers a viable solution. When applied to multi-agent systems, RL becomes multi-agent reinforcement learning (MARL), a rapidly growing field that introduces both new challenges and opportunities. One of the key advantages of MARL is its ability to reduce the action space compared to centralized approaches, helping to mitigate the curse of dimensionality commonly faced in such settings. However, this also presents new challenges, such as non-stationarity, credit assignment, scalability, and partial observability [34, 35].

A common way to formalize MARL systems is through Markov Games (MG), as introduced in [103]. A Markov Game generalizes the MDP by considering the joint actions of N agents. The corresponding tuple is defined as:

$$\langle N, \mathcal{S}, \mathcal{A}^{(i)}_{i=\{1,\dots,N\}}, P_{\{a_1,\dots,a_N\}}, R_{\{a_1,\dots,a_N\}}, \gamma \rangle \ .$$

The solution to the MG differs from that of an MDP because the optimal strategy for each agent depends not only on its own policy but also on the policies of other agents in the environment [34]. This interdependence introduces non-stationarity, which is further complicated by partial observability, transforming the problem into a partially observable MDP (PO-MDP) for each agent. Various strategies have been proposed to address these challenges. One popular approach is centralized training-decentralized execution (CTDE), where agents exchange information during training. Another approach, discussed earlier in this chapter, is to enable communication among agents, helping them overcome non-stationarity through integrated communication and learning.

In the following subsections, we will first introduce the CTDE framework and then discuss methods to foster collaboration among agents through communication over graph structures. This concept will be further explored in the next section, where we outline strategies for defining the graphical framework to effectively integrate communication and learning in a MARL environment.

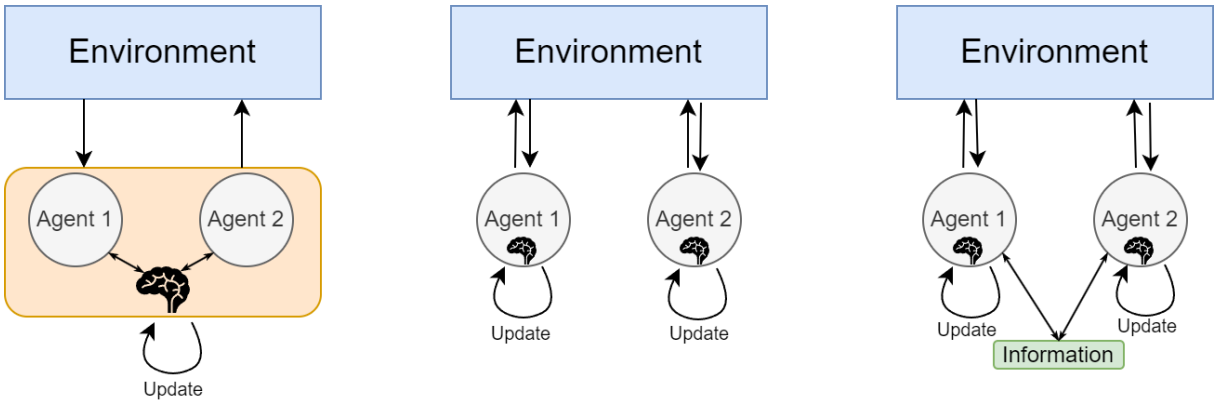


Figure 5.1: CTCE vs DTDE vs CTDE training schemes in MARL.

Centralized Training Decentralized Execution

The CTDE paradigm is the state-of-the-art approach for learning in multi-agent systems [35]. In CTDE, agents utilize shared computational resources or dedicated communication channels to exchange information during the training phase. This helps address non-stationarity by allowing agents to differentiate between their own actions and the collective actions of all agents in the system. CTDE stands in contrast to other training methods, such as centralized training-centralized execution (CTCE) and decentralized training-decentralized execution (DTDE). Fig. 5.1, inspired by [35], illustrates the key characteristics of these training schemes. In CTCE, a single centralized controller manages all agents, inherently solving the issue of non-stationarity. However, this approach lacks scalability and suffers from the curse of dimensionality, as complexity grows exponentially with the number of agents and actions. On the other hand, DTDE involves decentralized learning by individual agents, but it still struggles with scalability and non-stationarity. CTDE, in contrast, mitigates these issues by facilitating information exchange during training, which can include various forms of data sharing, such as policy parameters (e.g., *parameter sharing* [104, 105]).

5.3 Multi-Agent Network Optimization

Wireless communication networks are complex systems that require careful optimization of network procedures to meet predefined performance goals. MARL, with its inherent advantages, has emerged as a promising approach for solving a variety of network optimization problems. However, the practical application of MARL in real-world systems faces challenges, particularly regarding convergence, which remains an active area of research. This section highlights the use of graph structures as a powerful tool to mitigate non-stationarity in MARL systems by introducing a relational inductive bias into the decision-making process. By employing

GNNs as neural architectures for policy parameterization, the learning process benefits from the convolution of features over neighboring entities. The core idea behind this approach is that selecting the architecture with the appropriate bias for the problem at hand can significantly enhance cooperation among agents, allowing them to share local information with relevant peers to overcome partial observability.

To illustrate the potential of this approach, this section applies graph structures to solve a power control optimization problem, providing a practical example within the domain of radio resource management and network optimization in mobile radio networks. Special attention is given to capturing the interactions between neighboring agents by introducing innovative strategies for defining a graph-based framework that integrates communication and learning.

5.4 Literature Overview

The application of GNNs for optimizing wireless networks has garnered significant attention in recent literature. This interest stems from the inherent characteristics of GNNs, which offer scalable solutions, exhibit inductive capabilities, and, thanks to their permutation equivariance property, enhance generalization. Notably, these properties have been leveraged in studies such as [106], where GNNs are employed to model the dynamic structure of fading channel states, enabling the learning of optimal resource allocation policies in wireless networks. Another key area where GNNs have been extensively utilized is channel management in wireless local area networks (WLANs), as demonstrated by works like [107] and [108]. A particularly notable finding in [107] is the ability of GNNs to perform decentralized inference, making them a promising approach for the practical implementation of MARL systems. Similarly, GNNs have been applied to address power control optimization in wireless networks, as explored in works such as [109], [110], and [111].

Across the relevant literature, GNNs are deployed either as centralized controllers or as decentralized entities that model data-driven policies through feature convolution over graphs. However, despite the crucial role that graph structure plays in defining agent interactions, the impact of different graph formation strategies on achieving collective goals is a topic often overlooked. The choice of graph structure directly influences how communication occurs among distributed agents, which is essential for enabling effective cooperation. Numerous studies within the MARL domain highlight that communication is key to fostering multi-agent cooperation. For example, [112] argues that unrestricted information sharing among all agents in a distributed setting can hinder the learning process. To address this, they propose an attentional communication model that learns *when* communication should occur. Another significant work, [113], demonstrates how targeted communication—where agents learn both *what* messages to send and *whom* to address—can mitigate the issue of partial observability in multi-agent set-

tings.

In this chapter, we introduce graphs as communication-enabling structures for distributed optimization problems within MARL frameworks. Specifically, we explore how the relational inductive bias, introduced through message transformation (i.e., *what* to communicate) and message passing (i.e., *whom* to communicate with), deeply influences decision-making. Furthermore, we demonstrate that modeling edge weights using increasingly sophisticated strategies significantly enhances the agents' ability to cooperate and improves overall system performance. Finally, we propose a novel strategy for learning optimal edge weights, which shows strong inductive capability and surpasses all baseline strategies based on domain knowledge.

5.5 System Model

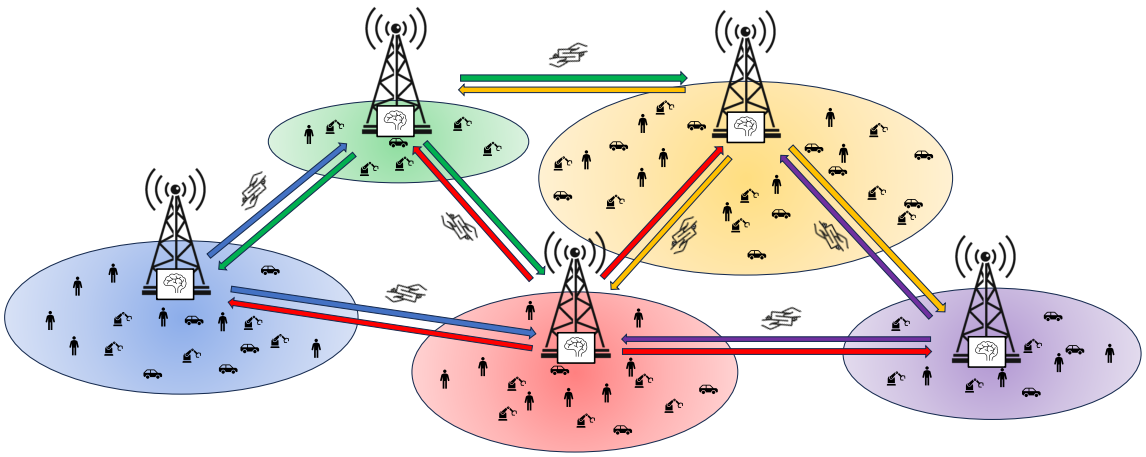


Figure 5.2: Base stations perform power control optimization in a decentralized manner by relying on a communication graph that enables the exchange of information exclusively among connected nodes.

The reference scenario depicted in Fig. 5.2 involves a set of serving nodes (base stations) in a wireless network. These nodes provide communication services to multiple UEs, which are categorized based on their distinct service and performance requirements. Specifically, three categories are considered, each with progressively stricter reliability requirements, defined in terms of bit error rate (BER). In this section, we describe the wireless network model, the requirements of the UEs, and the traffic distributions, along with the PO-MDP formulation.

Wireless Network Modeling

The network is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ represent base stations (agents)[1], and edges $\mathcal{E}$ represent virtual communication links between them. To define the graph structure, an adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is introduced, where each element $a_{u,v} \in A$ indicates the connectivity between nodes u and $v \in \mathcal{V}$.
A critical aspect of this analysis is the formulation of an appropriate graph structure for the set of base stations. In this context, we present four strategies for determining A.

Binary Edges

The first strategy involves binary edges, where $a_{u,v} \in \{0, 1\}$ indicates whether two nodes are connected, where

$$a_{u,v} = \begin{cases} 1 & \text{if } ||\mathbf{s}(u) - \mathbf{s}(v)||_2 < D \\ 0 & \text{otherwise} \end{cases}. \tag{5.1}$$

Here, $\mathbf{s}(u)$ denotes the position of node u in the 2D-space, $||\cdot||_2$ is the Euclidean distance, and $D \in \mathbb{R}$ is a connectivity threshold. This results in a symmetric adjacency matrix, and the corresponding graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is both *unweighted* and *undirected*.

Distance-Based Edges

A more informative approach is to define continuous edge values $a_{u,v} \in \mathbb{R}$, which are based on the physical distance between nodes. This is expressed as:

$$a_{u,v} \propto e^{-||\mathbf{s}(u)-\mathbf{s}(v)||}, \tag{5.2}$$

where $\mathbf{s}(u)$ denotes the position of node u in 2D space. This yields a *weighted* and *undirected* graph, as the physical distance between two nodes is a symmetric property.

Relation-Based Edges

This strategy defines edges based on the mutual interactions between nodes. Nodes are connected if their actions have a direct influence on each other. Evaluating this mutual interaction is complex and often requires tools such as directed graphical models [114, 115], causal inference [116], or, in the case of wireless networks, empirical models to quantify interference. This interaction can be expressed as:

$$a_{u,v} \propto \mathcal{I}(u|v), \tag{5.3}$$

[1] We use the terms base stations and agents interchangeably.

where $\mathcal{I}(u|v)$ represents the influence node v exerts on node u. Typically, $\mathcal{I}(u|v) \neq \mathcal{I}(v|u)$. In wireless communication, $\mathcal{I}(u|\cdot)$ could quantify the level of interference received by node u from its neighboring nodes in $\mathcal{G}$. This results in a *weighted* and potentially *directed* graph.

Learning-Based Edges

The final strategy involves an end-to-end learning approach for determining the adjacency matrix A. During the training phase of the agents, edge weights are learned alongside policy parameters. This is achieved using a secondary GNN, which dynamically assigns edge weights based on network topological features.
The process begins by establishing edge features, denoted as $\mathcal{F}_{u,v}$, for each directed edge $a_{u,v}$. These features are defined as:

$$\mathcal{F}_{u,v} = \{d_{u,v},\ \sin(\theta_{u,v}),\ \cos(\theta_{u,v})\}, \tag{5.4}$$

where:

- $d_{u,v} \triangleq \|u - v\|_2$ is the Euclidean distance between nodes u and v,
- $\theta_{u,v} \triangleq \arctan 2(v_y - u_y, v_x - v_x)$ is the angle between nodes u and v.

Next, an auxiliary graph $\mathcal{G}_f = (\mathcal{V}_f, \mathcal{E}_f)$ is introduced, where nodes $\mathcal{V}_f$ correspond to the edges of the original graph $\mathcal{G}$. The adjacency matrix for $\mathcal{G}_f$ is binary, where two nodes in $\mathcal{V}_f$ are adjacent if their associated edges in $\mathcal{G}$ share a common node. This is expressed as:

$$a_{u_f,v_f} = \begin{cases} 1 & \text{if } \mathbf{o}(u_f) = \mathbf{o}(v_f) \\ 0 & \text{otherwise} \end{cases}, \tag{5.5}$$

where $\mathbf{o}(u_f)$ denotes the node $o \in \mathcal{V}$ from which the edge associated with u_f originates.
Finally, the auxiliary GNN computes edge weights for the original graph $\mathcal{G}$, while a primary policy GNN uses these weights to optimize power control. Both GNNs are updated jointly during backpropagation, allowing for simultaneous learning of edge weights and policy parameters.

UEs' Requirements and Traffic Modeling

In the proposed system model, the process of generating users follows a Poisson cluster process (PCP). A PCP is a stochastic point process that represents the union of points generated by M independent homogeneous Poisson point processes (PPPs) centered around base stations in the Euclidean space $\mathbb{R}^2$. Formally, let Φ_{C_i} represent a homogeneous PPP centered at base station C_i with intensity $\lambda_{C_i} > 0$. This PPP generates a set of random points $\mathbf{s} \in \mathbb{R}^2$, denoted by $\mathcal{C}_{\mathbf{s},i}$.

Each user is classified into one of S distinct categories based on their BER requirements. Consequently, the i-th base station C_i is characterized by S intensity parameters, denoted as $\lambda_{C_i}^{(k)}$, where $k \in \{1, \ldots, S\}$ represents the category index. These intensity parameters account for the distinct user categories associated with each base station. The resulting PCP, denoted by $\mathcal{U}$, is then expressed as the union of all generated points:

$$\mathcal{U} = \bigcup_{\substack{i \in \{1,\ldots,M\} \\ k \in \{1,\ldots,S\}}} \mathcal{C}_{\mathbf{s},i}^{(k)}. \tag{5.6}$$

To satisfy the varying BER requirements of different user categories, an independent adaptive modulation and coding scheme (MCS) is employed for each category. This adaptive MCS adjusts the link's spectral efficiency, denoted by η, on a per-user basis to ensure that the required average BER is met for every k-th user category.

The relationship between the signal-to-noise ratio and the average bit error rate P_b for an uncoded M-QAM modulation scheme can be derived from the union bound on error probability. This yields a closed-form expression that depends on the minimum distance between signal constellation points, as given by [117]:

$$P_b \simeq \frac{1}{\log_2 L} \frac{L-1}{L} \operatorname{erfc}\left(\sqrt{\frac{|h_0|^2}{2 \cdot N}}\right), \tag{5.7}$$

where $L = \sqrt{M}$ represents the modulation constellation order, $2|h_0|$ denotes the minimum distance between signal constellation points, and N is the noise power.

5.6 PO-MDP Formulation

The MARL problem is modeled as a partially observable Markov decision process (PO-MDP), where agents gather local observations from the global environment. Since the optimization problem involves distributed agents working towards an optimal power-tuning configuration, and there are no temporal dependencies between the agents' actions, the problem is treated as a *stateless* PO-MDP. In this formulation, the state transition dynamics are independent of past states or actions. The corresponding PO-MDP tuple is defined as $\langle \mathcal{S}, \mathcal{A}, R_a \rangle$.

Observation Space

Each agent collects information on user traffic distribution over a fixed-size grid G, defined in polar coordinates and partitioned into bins, indexed by distance d and angle ϕ relative to the base station's position. This results in a state space of fixed dimensions, which can be represented as

a 3D tensor:

$$\mathcal{S} = \begin{bmatrix} \mathbf{u}_{d_1,\phi_1} & \dots & \mathbf{u}_{d_1,\phi_n} \\ \mathbf{u}_{d_2,\phi_1} & \ddots & \mathbf{u}_{d_2,\phi_n} \\ \mathbf{u}_{d_m,\phi_1} & \dots & \mathbf{u}_{d_m,\phi_n} \end{bmatrix} , \tag{5.8}$$

where $\mathbf{u}_{d,\phi} = (u^{(1)}_{d,\phi}, \dots, u^{(S)}_{d,\phi})$ is a vector representing the aggregated traffic for all user categories, and each $u^{(k)}_{d,\phi}$ is calculated as:

$$u^{(k)}_{d,\phi} = \sum_{l \in G_{d,\phi}} t^{(k)}_l , \tag{5.9}$$

where $t^{(k)}_l$ represents the traffic demand of the l-th user of category k in the bin indexed by (d, ϕ) within G.

Action Space

Each agent i is tasked with tuning its own transmission power $\mathbf{p}_i$, selected from a discrete set, based on its local observations. The action space is defined as:

$$\mathcal{A} = \{p_0, \dots, p_P\} , \tag{5.10}$$

where P represents the number of available power levels.

Optimization Problem

The objective guiding the distributed agents is to achieve an optimal solution for the power control problem. The set of agents aims to solve the following maximization problem:

$$\arg\max_{\mathbf{p}} \sum_{l \in G} \sum_{k} \eta^{(k)}_l(\mathbf{p}, B_l) B_l , \tag{5.11}$$

where $\eta^{(k)}_l(\mathbf{p}, B_l)$ represents the link spectral efficiency for the l-th user, considering all users within the reference area defined by G. The link spectral efficiency $\eta^{(k)}_l$ is a function of the perceived SINR, the service category k (each corresponding to different requirements of BER), and the available bandwidth for the l-th user B_l that is affected by the noise power. In this formulation, power constraints are inherently set by the action space definition (Sec. 5.6), as each agent (or base station) must operate within its maximum available power p_P. Additionally, B_l is determined by the specific scheduling mechanism and the total number of users connected to the serving cell of that user and considering a system bandwidth B. In this framework, we consider a full frequency reuse scenario, meaning that all users share the entire system bandwidth B without any dedicated frequency partitioning.

The objective function in (5.11) takes into account two critical factors: first, agents must choose an appropriate collective power configuration $\mathbf{p}$ to maximize the average link spectral efficiency. Second, the agents should perform mobility load balancing (MLB) to balance the number of users across different base stations. User assignments to base stations occur after executing action a, following a best-server criterion. Notably, optimizing the link spectral efficiency depends on the distribution of user categories across base stations, which have distinct BER requirements, leading to complex interference dynamics.

5.7 Graph Multi-Agent Reinforcement Learning

In the considered scenario, centralized training with decentralized execution (CTDE) is employed, alongside parameter sharing. This method enables agents to share their individual trajectories with a centralized entity during training, facilitating the learning of a shared policy.
Policy derivation relies on policy gradient methods. Specifically, for the numerical experiments detailed in the following sections, the REINFORCE algorithm [37, 38] is used. As described in Sec. 2.1, REINFORCE provides an approximation of the state distribution u and the value function Q, as defined in Eq. (2.7), using Monte Carlo sampling. Notably, REINFORCE follows a model-free, on-policy approach, where the summation over states s and actions a can be naturally substituted by an averaging procedure over the target policy π:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[Q_{\pi}(S_t, A_t)\nabla\pi(a \mid S_t, \boldsymbol{\theta})\right] . \tag{5.12}$$

Through algebraic derivation [36], Eq. (5.12) yields the following update rule for REINFORCE:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \underbrace{\nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})\, R(\tau)}_{\nabla J(\boldsymbol{\theta})} , \tag{5.13}$$

where $R(\tau)$ represents the return for a trajectory τ sampled over policy π. In the scenario considered, the MDP is stateless, meaning each episode corresponds to a single action step.
While the policy is trained centrally, during execution, each agent has access only to local information, and all agents operate in a distributed manner with implicit coordination. By parameterizing the policy π with GNNs, a fully decentralized execution strategy is enabled. This approach allows agents to transform and aggregate local features from neighboring agents via message passing. The GNN architecture used in the numerical experiments is the local k-dimensional GNN (Local k-GNN) [118], whose update function can be expressed as follows:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_1^{(l)}\, \mathbf{h}_v^{(l)} + \mathbf{W}_2^{(l)}\, \mathbf{AGG}\left(\left\{e_{u,v}^{(l)} \cdot \mathbf{h}_u^{(l)},\ \forall u \in \mathcal{N}_v\right\}\right)\right) , \tag{5.14}$$

where $e_{u,v}^{(l)}$ represents the edge weight between node u and node v, which is computed using

one of the strategies presented in Sec. 5.5, while $\mathbf{W}_1^{(l)}$ and $\mathbf{W}_2^{(l)}$ are learnable matrices. As demonstrated by the numerical results in the subsequent sections, modeling edge weights using different strategies (i.e., introducing biases on *what* to communicate through message transformation, and *to whom* to communicate via message passing) has a significant impact on the agents' ability to learn cooperative behaviors, ultimately enhancing overall performance.

5.8 Simulation and Numerical Results

To validate the proposed framework and assess the impact of different graph modeling strategies on learning cooperative behavior, a series of numerical experiments were conducted in a simulated network scenario. The setup consists of 11 base stations with randomly generated traffic, as outlined in Sec. 5.5. The system bandwidth is set to $B = 60$ MHz, the carrier frequency to $f_c = 3.7$ GHz, and the channel model follows the 3GPP UMa path loss model [119]. Four graph modeling strategies discussed in Sec. 5.5 are evaluated, using a baseline strategy that employs REINFORCE combined with a classical DNN-based policy parameterization.
Specifically, the "relation-based edges" strategy—introduced in general terms—has been evaluated here using a mutual interference criterion. This criterion considers an average-to-worst-case scenario, where the serving cell transmits at average power, while the interfering cell transmits at maximum power. Mutual interference is significantly influenced by the distribution of users across different service categories. Base stations that primarily serve users with more demanding requirements are more susceptible to inter-cell interference compared to those serving other user groups. For all training and inference tests, the number of service categories, S, is set to 3, where the the BER requirements for each category are set to 10^{-1}, 10^{-3}, and 10^{-5}.

Training Performance

Fig. 5.3 presents the training performance of the simulated network environment. It shows the evolution of the average reward over the training epochs for the four graph models and the baseline, highlighting their respective performance. A total of 30 training instances were conducted, and the results are presented as mean rewards with 99% confidence intervals. As the figure indicates, the use of GNN-based policy parameterization results in faster convergence and overall better performance compared to the DNN-based policy. Since both methods employ CTDE with parameter sharing, the results emphasize the importance of enabling agents to communicate their local features to their neighbors, which significantly enhances cooperation and overall performance. Furthermore, the figure shows that the choice of graph structure plays a critical role in performance. The unweighted graph with binary edges, which cannot differentiate between more or less relevant neighbors, achieves only mediocre results. In contrast, graph structures that embed contextual information—such as physical proximity (orange curve)

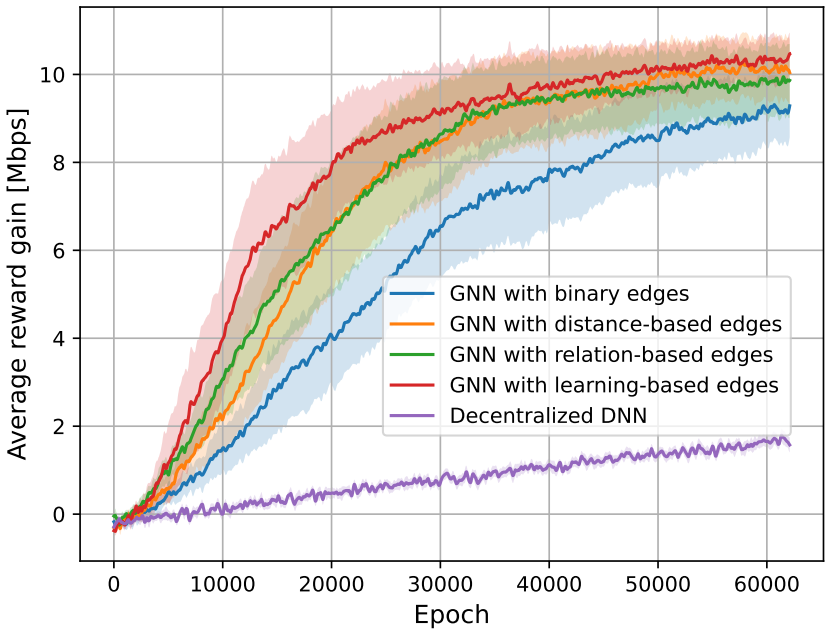


Figure 5.3: Training performance as a function of the number of training epochs.

or mutual interference (green curve)—outperform the simpler models. This finding highlights the advantage of incorporating domain knowledge into the problem formulation, driving the agents' collective behavior towards a better solution in a shorter time. Lastly, the graph with learnable edge weights (red curve) stands out as the most effective approach, as it learns the optimal edge weighting and policy parameterization in an end-to-end fashion by leveraging the geometrical features of the environment, as described in Sec. 5.5.

Generalization Tests

A distinctive feature of GNNs is their strong generalization capability, attributed to their inherent permutation equivariance (Sec. 2.3). This section presents and discusses the inductive capability of the proposed models. As in the training phase, results are based on 30 independent runs, with the mean score reported alongside 99% confidence intervals. The values are normalized with respect to the rewards achieved by the learnable edges-based GNNs trained on larger networks for the same number of epochs.

Fig. 5.4 illustrates the performance of the learned models when deployed in networks of increasing size. It is evident that all GNN models maintain stable or slightly improved performance as the network grows in size. This trend is linked to the increasing complexity of the learning task, which becomes more challenging as the network size expands. These results suggest that training on smaller scenarios effectively scales to larger, unseen environments. Notably, the GNN with learnable edges demonstrates superior performance, indicating not only better

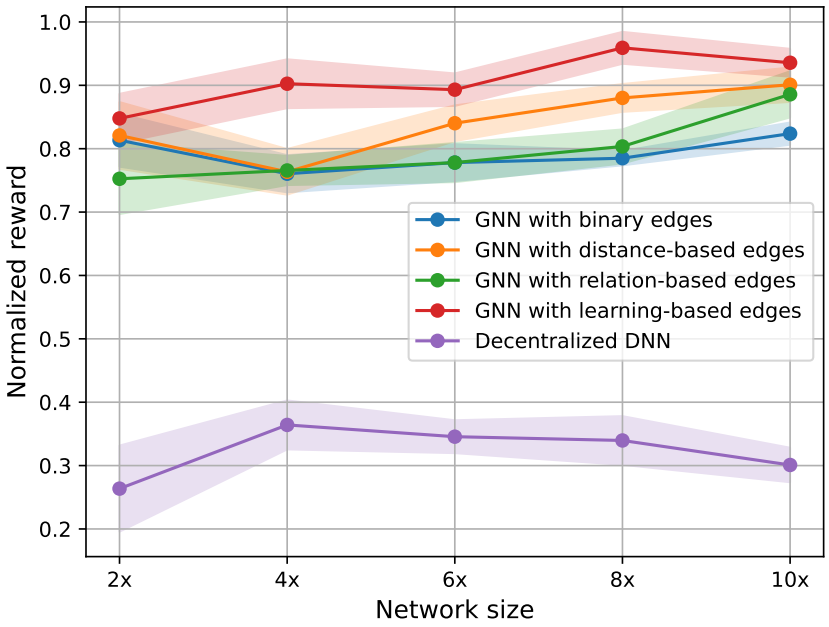


Figure 5.4: Inference test – network of increasing size.

policy learning but also better generalization to larger networks.

Finally, Fig. 5.5 presents the performance of the learned models when tested on networks with increasingly distinct traffic patterns from those seen during training. In the simulation, traffic is modeled using a Poisson cluster process (PCP), with each base station linked to three λ rates corresponding to each user category. The x-axis measures the difference in traffic patterns using cosine similarity between the vectors of λ rates, where the vector dimension is three times the number of agents in the system. As in the previous case, results are normalized with respect to the rewards of the learnable edges-based GNNs trained under increasingly different traffic conditions. Despite a slight performance drop with varying traffic patterns, all GNN models show strong generalization ability. Again, the GNN with learnable edges delivers the best performance, underscoring its robustness in diverse traffic conditions.

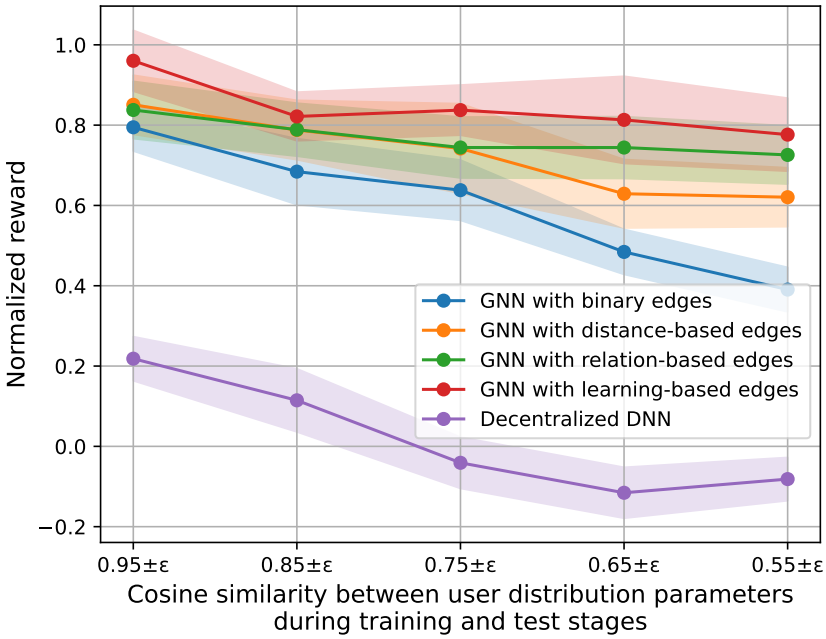


Figure 5.5: Inference test – varying user distribution.

Chapter 6

Deep Learning and 5G Architectures for Industrial IoT

The advent of 5th generation (5G) cellular technology has marked the beginning of a new era in connectivity and innovation [120]. Serving as the backbone for the IoT, 5G networks are poised to transform how data is collected, processed, and applied across multiple sectors and industries [121], [122], [123]. In this context, 5G technology empowers the industrial internet-of-things (IIoT) by enabling continuous equipment monitoring, with real-time data collection from sensors and machinery [124], [125], [126]. This data-rich environment, coupled with advanced data analytics and deep learning (DL) algorithms, enables industries to optimize operations, extend equipment lifespan, reduce downtime, and mitigate safety risks from potential failures [127]. As described in [124] and [125], the 5G Alliance for Connected Industries and Automation (5G-ACIA) categorizes industrial use cases into several application areas:

- **Factory Automation**: Encompasses use cases for the automated control, monitoring, and optimization of processes and workflows within a factory setting.
- **Process Automation**: Focuses on controlling production processes and managing materials, improving efficiency, safety, and energy use.
- **Human-Machine Interface (HMI)**: Involves use cases where human operators interact with production systems.
- **Logistics and Warehousing**: Covers transportation and storage of goods and materials in industrial environments.
- **Monitoring and Maintenance**: Pertains to passive monitoring of industrial processes and equipment for maintenance purposes.

Table 6.1: Association between application areas (rows) and industrial use cases (columns) [124], [125].

	MC	CC	MCP	MR	MWSN	RAM	AR	CLPC	PM	PAM
Factory automation	×	×		×	×					
Process automation				×	×			×	×	×
HMI			×				×			
Logistics and warehousing		×		×						×
Monitoring and maintenance				×	×	×	×			

Each application area corresponds to a set of use cases. The relationship between application areas (rows) and specific use cases (columns) is shown in Table 6.1.
The 5G-ACIA identifies several key use case categories for IIoT applications:

- **Motion Control** (MC in Table 6.1): These systems control the movement of industrial machinery components with precision. Motion control systems, often implemented as closed-loop control systems, cyclically collect sensor data and issue actuator commands. An example is in automotive production, where conveyor synchronization for the body-chassis marriage requires reliable communication.

- **Control-to-Control** (CC in Table 6.1): This category ensures safe interactions between machinery and human operators. It includes two sub-cases:

  - *Local Control-to-Control*: Involves devices with separate controllers interacting to perform tasks, like shuttles in packaging machines communicating positions over 5G interfaces. Failed communication can lead to machine shutdowns.

  - *Remote Control-to-Control*: Involves devices that typically operate autonomously but need remote control for service or maintenance, such as remote control in assembly lines.

- **Mobile Control Panels** (MCP in Table 6.1): Related to configuring, monitoring, and maintaining machines or production lines.

- **Mobile Robots** (MR in Table 6.1): Mobile robots, used for tasks like object transportation in workshops, require precise interaction with their environment. Advanced industrial robots use virtualized control systems, enabling flexible reconfiguration and reducing deployment costs. AGVs, a sub-case, transport goods across factory floors. These vehicles can be remotely controlled via wireless links and equipped with obstacle detection, allowing for immediate response to potential issues.

- **Massive Wireless Sensor Networks** (MWSN in Table 6.1): Monitor industrial equipment and environmental conditions like temperature and vibration.
- **Remote Access and Maintenance** (RAM in Table 6.1): Refers to remote machine communication for maintenance purposes, such as inventory management or extracting event logs.
- **Augmented Reality** (AR in Table 6.1): Augmented reality enables remote monitoring and operator assistance in complex environments, like nuclear plants, via specialized equipment.
- **Closed-Loop Process Control** (CLPC in Table 6.1): Involves using sensor data to control actuators and prevent accidents. Two sub-cases include:
  - *Process Monitoring* (PM in Table 6.1): Focuses on gathering environmental data for process monitoring. Reliability is a key requirement here.
  - *Plant Asset Management* (PAM in Table 6.1): Utilizes sensor data to assess the health of industrial assets, such as pumps or valves, detecting malfunctions in real-time.

This chapter will conduct a holistic performance analysis of 5G IIoT networks in safety-critical scenarios. As an illustrative example, in the context of "Monitoring and Maintenance," we explore the case of estimating the RUL of critical assets. Consider AGVs transporting hazardous liquids, where failure could pose a risk to workers. These events can be anticipated using data-driven RUL estimation. In 5G-enabled systems, AGVs collect real-time sensor data and transmit it to a server, where DL methods predict failures. This enables preventative actions, such as stopping the AGVs if a failure is foreseen, preventing liquid spills.

6.1 Literature Overview

The performance of 5G networks has been widely explored in the literature. Simulation tools are indispensable for studying the behavior of complex wireless systems, such as cellular networks. For instance, a system-level simulator is developed in [128] to investigate 5G networks, and it is further expanded in [129] to study ultra-reliable low latency communication (URLLC) scenarios, focusing on key metrics like latency, reliability, and throughput. Similarly, *ns-3* [130], a discrete event network simulator, enables the simulation of various network types. The module introduced in [131] allows the simulation of 5G mmWave networks, analyzing several KPIs. In addition, custom simulators, such as the one in [132], have been developed to study URLLC use cases in IIoT scenarios. While these works evaluate 5G systems primarily through

KPIs and occasionally refer to requirements like those outlined by 5G-ACIA in [124], they do not focus on experimental industrial use cases with specific requirements.

With the advancements in 5G devices, empirical evaluations of 5G New Radio (5G NR) performance are emerging, as seen in [133, 134]. The study in [135] assesses 5G performance for industrial automation but considers only generic requirements and use cases, not real-world applications.

In contrast, the works in [136] and [137] explore IIoT scenarios where a 5G network enables communication between an AGV and a remote programmable logic controller (PLC) that controls the AGV's movements. [136] evaluates system performance in terms of deviation from planned trajectories and energy consumption, while [137] seeks to predict AGV malfunctions by analyzing network traffic data. Despite the innovative approaches, both studies exhibit certain limitations. First, they only evaluate performance in terms of deviation from the planned trajectory, neglecting the application's specific requirements and failing to break down the impact of network, actuation, and inference times on performance. Additionally, they do not assess the influence of multiple AGVs on system performance.

Reliable remaining useful life (RUL) estimation is another critical component in industrial environments, with the popularity of data-driven approaches rapidly growing. These techniques use historical sensor data to detect potential hazards by analyzing signal patterns and establishing RUL through inferred data correlations and causal relationships. Deep learning methods are increasingly employed in this domain. For instance, autoencoders (AEs) are utilized in RUL estimation to compress complex features into main components, followed by predictions using DL networks [138]. recurrent neural networks (RNNs), specifically deep long-short term memory (LSTM) networks, have also been used for low-error prediction models, as demonstrated in [139, 140]. More recently, CNNs have gained traction; in [141], multiple sensors and time windows are used to predict the RUL of industrial engines. Combining these approaches can further enhance prediction accuracy [142, 143].

Building on this background, this chapter introduces a safety-critical scenario in which we develop a DL-based pipeline for RUL estimation, leveraging real sensor data from experiments conducted in a pilot production plant. We investigate the performance of 5G networks under various configurations of 5G NR and 5G core network (5CN), focusing on key factors such as the number of AGVs, bandwidth, operating frequency, and the number of gNBs on RTT. A holistic analysis is performed on the system, combining the 5G network with the DL-based pipeline, addressing RTT requirements, and considering the inter-dependencies between communication networks and data-driven applications in IIoT settings.

6.2 Data-Driven RUL Prediction Pipeline

This chapter provides a comprehensive performance analysis of 5G IIoT networks in safety-critical environments. As an example, we examine the estimation of the RUL of vital assets. Specifically, consider AGVs transporting hazardous liquids, where a malfunction could endanger workers. These risks can be mitigated by using data-driven RUL predictions. In 5G-enabled systems, AGVs continuously gather real-time sensor data and send it to a server, where deep learning techniques are applied to predict potential failures. This allows for proactive measures, such as halting the AGVs before a failure occurs, preventing hazardous spills.

RUL Estimation as Binary Classification Problem

RUL prediction is a critical task that aims to estimate the amount of time until a machine or system fails [144]. RUL prediction can be formulated as a binary classification task, where the objective is to predict whether a machine or system will fail within a certain time horizon. Within this formulation, sensor data are used to train DL models that can accurately classify machines into different RUL classes. In the considered scenario, the classes are two: one for representing the correct functioning of the system, the other for anomalous situation leading to liquid falls from the AGV.
In binary classification settings, it is necessary to define the concept of margin, inherently bounded to the problem formulation. The margin is a parameter of the system that corresponds to the number of samples preceding the anomaly that are classified as anomalous. In this scenario, if the margin is m and sampling frequency is 10 Hz, then the algorithm can predict the anomaly at most $m - 1$ tenth of seconds in advance. In the time series, the last m samples will be labeled as anomalous, while the others will be associated with the correct functioning of the system.

Deep Learning-based Pipeline for RUL prediction

To perform RUL estimations in binary classification settings, we implemented DL-based pipeline involving two main components: a DL model and an optimized threshold. The model is trained on data collected during the experimental campaign to predict the RUL of the system. In general, the raw output of a model trained for binary classification is produced by a sigmoid and it is rounded to a binary value using the default threshold of 0.5. However, in our pipeline this threshold is further fine-tuned via an iterative optimization algorithm, which aims to find the optimal threshold to be applied on the DL model output which minimizes the cost determined with a given cost model.
In the pipeline, the dataset should be partitioned in 4-folds for training, optimization, and testing

purposes. Specifically, the partitions required are: 1) a training set to train the DL model, 2) a validation set to assess the DL model performance at training time, 3) another validation set to optimize the threshold, and 4) a test set to evaluate the overall performance.
Several DL algorithms are selected and tested, including logistic regression (LR), deep neural networks (DNNs), autoencoders (AEs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), and vanilla transformers (VTs) [145–148]. The training code is detailed and presented on Github [1], including the set of hyper-parameters tested. An optimal threshold over the AGV axial acceleration is considered as baseline model for this task, since in this use case abrupt braking is often associated to fall events.

6.3 System Model

In this section, the network system model implemented in the simulator is presented. We illustrate the network architecture, and the deployment, channel, and traffic models.

Network Architecture

Several architectures are considered, with different 5G NR and 5CN setups. The considered 5G NR configurations are the following:

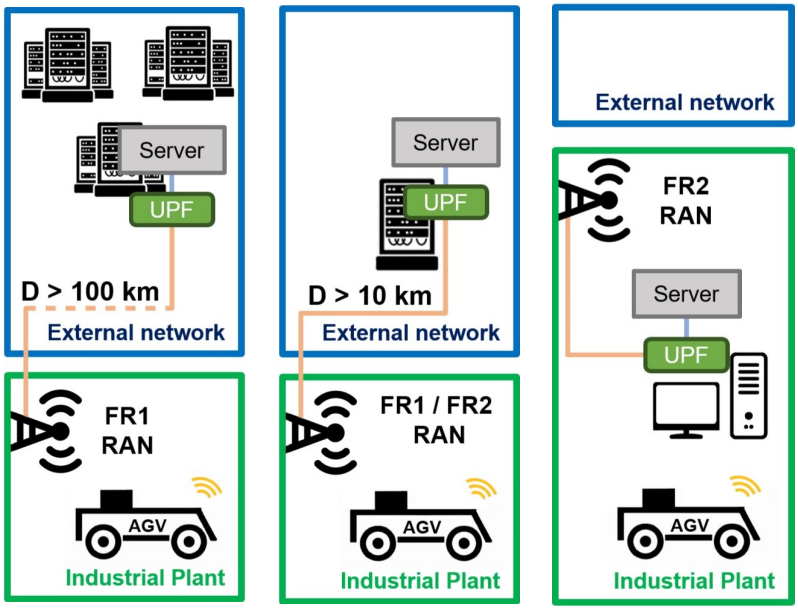
1. RAN operating in FR1;
2. RAN operating in FR2.

We assume that RAN is always deployed inside the industrial plant. The impact of different number of gNBs in each case is explored in following sections.
At the same time, we consider various configurations of 5CN, varying in the locations of user plane function (UPF) and the application server and their distance D from the industrial plant:

1. non public network (NPN) on-premise: this configuration features a completely private network, with 5CN deployed within the factory ($D = 0$ m);
2. NPN on-net: the 5CN is hosted at operator's premises, and a dedicated pool of resources is allocated to the client. 5CN is located up to tens of kilometers away ($D > 10$ km);
3. public network (PN): the used 5CN is the public one, there it can be hundreds of kilometers away ($D > 100$ km).

Different 5G NR and 5CN configurations lead to different performance in terms of RTT. We consider four different architectures, and from now on they are referred to as follows:

[1] Code: https://github.com/Lostefra/5G-IoT-AGV-RUL-prediction



(a) Arch. 1. **(b)** Arch. 2 and 3. **(c)** Arch. 4.

Figure 6.1: Representation of the four considered architectures.

1. Architecture 1: PN with RAN operating in FR1;
2. Architecture 2: NPN on-net with RAN operating in FR1;
3. Architecture 3: NPN on-net with RAN operating in FR2;
4. Architecture 4: NPN on-premise with RAN operating in FR2.

The four different architectures are pictured in Fig. 6.1.

Deployment Model

Before describing the deployment model, it is worth mentioning that, although the data gathering has been conducted using a single AGV, in the following sections we assume to have several AGVs. Furthermore, the simulated production plant is larger than the real one since the latter is too small to accommodate the intended number of AGVs. The industrial plant is represented as a rectangular cuboid whose dimensions are length (l), width (w), and height (h), as shown in Fig. 6.2. Production machines are modelled as cubes of side s_c positioned to maintain a given inter-machine distance d (measured from the center of the lower base), and they act as obstacles for communications between AGVs and gNBs. Within the factory, N AGVs are deployed in areas not occupied by the obstacles. The RAN is composed by one or two gNBs, located in positions determined by the output of Alg. 3 and operating in FR1 or FR2, depending on the architecture. If multiple gNBs are present, the total available bandwidth is equally split

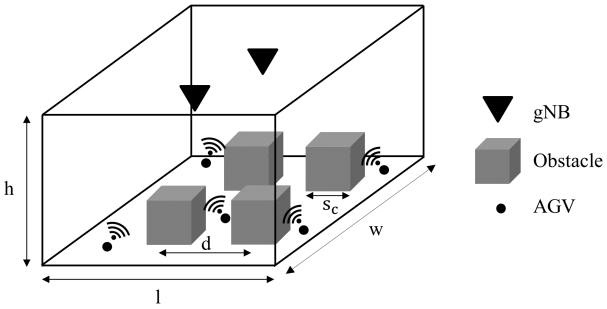


Figure 6.2: Reference industrial scenario with 2 gNBs.

Algorithm 3 gNBs placing algorithm

Variables:
- x: longest side of the factory plant
- y: shortest side of the factory plant
- N_G: number of gNBs, $N_G \in \{1,2\}$
- coordinates$_j$: position of gNB$_j$, $j \in \{1, \ldots, N_G\}$

Start:
$step\_x \leftarrow \frac{x}{N_G \cdot 2}$
for $j \in \{1, \ldots, N_G\}$ **do**
$x_j \leftarrow step\_x \cdot (2j - 1)$
$y_j \leftarrow \frac{y}{2}$
coordinates$_j \leftarrow (x_j, y_j)$
end for

between them, whitout considering frequency reuse. Each AGV communicates in both uplink and downlink with gNBs, whose traffic model is described in the Sec. 6.3.

Channel Model

The channel model considered is detailed in 3GPP technical report (TR) 38.901 [149]. In particular, the model proposes four indoor factory (InF) scenarios, depending on the density of obstacles and the height of transmitters and receivers. Each of them is characterized by different path loss (PL) and different log-normal shadowing fading. PL and shadowing, in turn, depends on having line-of-sight (LoS) or non line-of-sight (NLoS) condition between AGVs and gNB. This property is verified geometrically in our simulator, observing if the line that joins AGV and gNB intersects any obstacle. After determining PL, it is possible to evaluate the signal quality. The figure of merit we consider is signal-to-noise-ratio (SNR), expressed as follows:

$$\text{SNR} = \frac{P_{\text{RX}}}{P_N}, \tag{6.1}$$

where P_{RX} is the received power and P_N is the noise power. P_{RX} can be expressed as:

$$P_{RX} = \frac{P_{\text{TX}} \cdot G_{\text{TX}} \cdot G_{\text{RX}}}{\text{PL} \cdot \text{SH}}, \tag{6.2}$$

where P_{TX} is the transmit power, G_{TX} is the transmission gain, G_{RX} is the reception gain, PL is the path loss, and SH is the log-normal shadowing component. P_N can be expressed as:

$$P_N = k_B \cdot T \cdot B, \tag{6.3}$$

where k_B is the Boltzmann constant, T is the noise temperature, and B the bandwidth used by the gNB. If two gNBs are present, it is clear that, since each gNB uses half of the available bandwidth, P_N will be lower.

SNR determines if a data plane physical (PHY) protocol data unit (PDU) is correctly received and which is the modulation order used by the transmitter.

Traffic model

The traffic model implemented in this work emulates the behavior of AGVs within the reference scenario: each AGV periodically sends information to the application server in the format described in Sec. 6.6, while the server sends a potential "failure alert" message only when the DL-based pipeline predicts a liquid fall. Therefore, the transmission of messages from the server to the AGV is aperiodic and the distribution of this process is determined by the considered DL-based pipeline and by the path travelled by the AGV. In the simulator, we consequently modeled uplink traffic as periodic, and downlink traffic as Gaussian distributed with mean μ_{DL} and standard deviation σ_{DL}.

6.4 5G-NR Simulation Setup

In this section, we present the 5G NR compliant network simulator that has been developed. It is worth emphasizing that in this section, we refer to UEs instead of AGVs, since the implemented scheduler is independent of specific device types. We begin by introducing the 5G NR framework, followed by an overview on clustering of UEs, which determines the gNB serving them, and we conclude by explaining the implemented scheduler. The simulator implements a multi gNB system, an advanced channel model, and a scheduler.

5G-NR framework

We start this subsection by introducing the time-frequency structure determined by the OFDM waveform. In the frequency domain, we transmit on a carrier frequency f_c using a bandwidth B, and a subcarrier spacing (SCS) Δf. The bandwidth B is partitioned into n_{RB} RBs, with each consisting of 12 OFDM subcarriers, in particular:

$$n_{\mathrm{RB}} = \left\lfloor \frac{B}{12\Delta f} \right\rfloor, \tag{6.4}$$

It is worth mentioning that the available bandwidth B is equally split among available gNBs.
In the time domain, OFDM symbols are organized into slots of 14 OFDM symbols each. However, since Rel. 15, in order to reduce latency, it is allowed to transmit over fractions of slots, the so called "mini-slot" transmission. In this simulator, we used mini-slots of 7 OFDM symbols each, and we used it as scheduling unit (SU) in both control plane and data plane.
We implemented the messages used in [132, 150], which are:

- physical uplink control channel (PUCCH): used by UEs when they ask to the gNB resources for their uplink transmission. It occupies 1 SU and 1 RB;
- physical downlink control channel (PDCCH): used by gNB when it informs the UEs which resources they can use, if any, for uplink or downlink transmission. It occupies 1 SU and 1 RB;
- physical uplink shared channel (PUSCH): used by UEs to transmit data plane PHY PDU. It occupies at least 1 RB and 4 OFDM symbols;
- physical downlink shared channel (PDSCH): used by gNB to transmit data plane PHY PDU to UEs. It occupies at least 1 RB and 4 OFDM symbols;
- hybrid automatic repeat request (HARQ): used to notify the sender regarding the outcome of a PUSCH or a PDSCH transmission. It occupies 1 RB and 2 OFDM symbols.

The time needed to send PUSCH/PDSCH plus the reception of their correspondent HARQ is exactly 1 SU, assuming that 1 OFDM symbol is needed for the radio to switch from transmission to reception mode. It is important to note that this is done assuming that we are using half-duplex devices. The duration of 1 SU is fixed in terms of OFDM symbols but variable in terms of milliseconds because the OFDM symbol duration depends on SCS, which, in turn, affects the SU duration.

Algorithm 4 Clustering algorithm

Variables:
- UE_i: i-th UE, $i \in \{1, \ldots, N\}$
- gNB_j: j-th gNB, $j \in \{1, 2\}$
- $\text{SNR}_{i,j}$: SNR perceived by UE_i from gNB_j
- $\text{UE}_{i,j}$: UE_i is associated to gNB_j
- ξ: maximum imbalance factor between two clusters, $\xi \in [0, 1]$
- n_j: number of UE associated to gNB_j
- x: index associated to the gNB with more UEs
- y: index associated to the gNB with less UEs

Start:

```
/* compute all SNR_{i,j} */
for j ∈ {1,2} do
    for i ∈ {1,...,N} do
        compute SNR_{i,j}
    end for
end for

/* assign UEs to gNBs */
for i ∈ {1,...,N} do
    if SNR_{i,1} ≥ SNR_{i,2} then
        assign UE_{i,1}
    else
        assign UE_{i,2}
    end if
end for

/* perform load balancing */
while n_x > N · (ξ + 1) / 2 do
    find UE_{i,x} such that SNR_{i,y} ≥ SNR_{z,y}, ∀UE_{z,x}
    UE_{i,x} → UE_{i,y} /* associate UE_i to the other gNB */
end while
```

Multi-gNB management and Clustering

In this work, we compare the performance of a single gNB system with that of a multi-gNB. Specifically, we address the case in which the multi-gNB system comprises two gNBs. If there is only one gNB, it is placed at the center of the ceiling of the production plant, while if there are two gNBs, we place them in the centers of two rectangles of the same size that partition the production plant, using Alg. 3.

In a multi-gNB system clustering has to be performed: UEs are assigned to one of the gNBs based on their SNR. The clustering algorithm (Alg. 4) tries to maximize the SNR of each UE, while guaranteeing a certain level of balance among clusters. For example, when the maximum imbalance factor ξ is equal to 0.2, each cluster must have at maximum 60% of UEs and at minimum 40% of UEs.

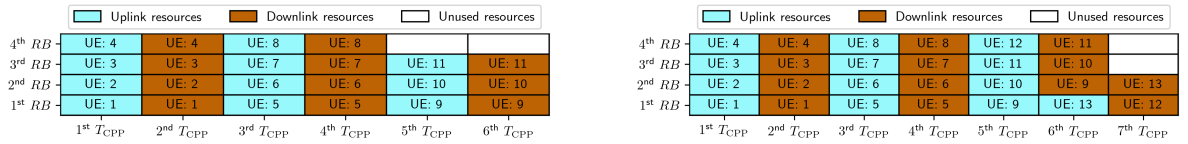
This algorithm is executed only once in the initialization phase of the system. It is worth mentioning that the simulator operates with a time basis of 1 OFDM symbol, which corresponds to the radio switch time, as shown in Sec. 6.4. In a real system, the time needed for AGVs to move from an area with good coverage to another with bad coverage, hence necessitating a

handover, can be in the order of tens of seconds or even minutes. Therefore, the simulator is not suitable to study the impact of handover on system performance: the implementation of a run-time handover algorithm and an AGV mobility model is beyond the scope of this work.
In Alg. 4, we firstly compute the SNR between each UE and each gNB. Subsequently, we assign each UE to the gNB that provides the best SNR, neglecting shadowing effects. If the cluster imbalance exceeds the maximum allowed, it is necessary to re-associate some of the UEs. In particular, considering gNB_x as the gNB with the higher number of UEs and gNB_y as the one with the fewest, the UEs currently associated with gNB_x that exhibit the highest SNR with respect to gNB_y are selected for re-assignment to gNB_y. It is worth mentioning that B is equally splitted among the two gNBs.

5G-NR scheduler

The communication is managed by schedulers, one per gNB, and they work independently from each other. Since the available bandwidth B is equally split among gNBs, and their bandwidths do not overlap, the scheduler does not have to handle interference. Each scheduler allocates resources to the UEs assigned to their gNB. Remarkably, assignments are performed by the clustering algorithm described in Section 6.4.
We implemented a dynamic scheduler. The time axis is organized into groups of 8 SUs, referred to as T_{CPP}, which represents the control plane periodicity. These groups are further subdivided into two sets of 4 SUs each: the first 4 SUs are used for control plane messages and are needed to provide resource allocation to the UEs, while the second 4 SUs are used for data plane messages and so for the effective transmission of information. In some instances, the number of RBs may not be sufficient to ensure that all the UEs can request and then receive assignment of resources in a single T_{CPP}. To address this, we concatenate multiple T_{CPP}, each one dedicated to a clearly defined set of UEs, until we serve them all in the control plane: we call this time interval T_{SRP}, representing the scheduling request periodicity, that indicates how often control plane resources are associated to an UE. Even if control plane resources are allocated every T_{SRP}, an UE might not have enough data plane resources to send all its bytes of information and in this case, other resources will be allocated in the subsequent T_{SRP}, unless a packet is discarded due to exceeding the maximum allowed latency (indicated as Q_{DL} for downlink transmissions and Q_{UL} for uplink ones).
The allocation of control plane resources is predetermined and it is static. Each UE has prior knowledge regarding the resources to be used for PUCCHs transmission and PDCCHs reception. The 4 SUs dedicated to control plane are used as follows:

- first SU is used for PUCCHs transmission;
- second SU is used by the gNB for processing the received PUCCHs;



(a) Control Plane resource allocation when the number of needed T_{CPP} is even. $n_{\text{RB}} = 4$ and $n = 11$.

(b) Control Plane resource allocation when the number of needed T_{CPP} is odd. $n_{\text{RB}} = 4$ and $n = 13$.

Figure 6.3: Depiction of control plane resource allocation process described in Alg. 5 showing the different approaches depending on the number of T_{CPP} needed. n indicates the number of AGVs while n_{RB} indicates the number of RBs.

- third SU is used for PDCCH transmission;
- fourth SU is used by UEs for processing the received PDCCH.

In the control plane part of a T_{CPP}, one RB is used for one PUCCH/PDCCH couple, having in total n_{RB} PUCCH/PDCCH couples in a T_{CPP}. For a single UE, uplink and downlink communication requires one PUCCH/PDCCH couple each, that means one RB each in the control plane, and these RBs might be in different T_{CPP}s. Since downlink transmissions do not require PUCCH messages, some resources are not utilized, but we have no alternative due to half-duplex nature of UEs. In Alg. 5, we introduce the algorithm used to allocate control plane resources, whose primary objective is to efficiently allocate control plane resources to UEs. We do not consider the trivial case in which only one T_{CPP} is needed to allocate all the resources. This allocation is designed to achieve, for each UE, a closely scheduled downlink after an uplink, thereby minimizing the RTT. Fig. 6.3 shows the control plane resource allocation.
Concerning the data plane, the scheduler allocates resources following some policies. The considered policies are the following, and they are applied in the order presented:

- **prioritization of downlink traffic:** given the importance of the information carried in the downlink direction related to potential faults, this traffic flow is prioritized over the uplink;
- **fairness first (FF)**: to maintain fairness, a minimum portion of data is served for each UE for each traffic flow direction. If there are still resources available, they are allocated to the remaining part of UEs' data;
- **first in first out (FIFO)**: the users are served based on a FIFO criterion.

Algorithm 5 Control Plane static resource allocation

Variables:
- n_{RB}: number of RBs
- n: number of UEs
- n_{UL}: number of UEs allocated in uplink
- n_{DL}: number of UEs allocated in downlink
- $\text{UL}_{i,u,v}$: PUCCH/PDCCH couple for uplink of UE_i in the u-th RB in the v-th T_{CPP}
- $\text{DL}_{i,u,v}$: PUCCH/PDCCH couple for downlink of UE_i in the u-th RB in the v-th T_{CPP}
- t: integer variable indicating the current T_{CPP}, $T_{\text{CPP}} \geq 2$
- b: integer variable indicating the current RB

Uplink and downlink allocation:

function ALLOCATE_UL_DL(t):
  for $b \in \{1, \ldots, n_{\text{RB}}\}$ **do**
    if $t \,\%\, 2 == 1$ **then**
      $n_{\text{UL}} \leftarrow n_{\text{UL}} + 1$
      allocate $\text{UL}_{n_{\text{UL}},b,t}$
      if $n_{\text{UL}} == n$ **then**
        break
      end if
    else
      $n_{\text{DL}} \leftarrow n_{\text{DL}} + 1$
      allocate $\text{DL}_{n_{\text{DL}},b,t}$
      if $n_{\text{DL}} == n$ **then**
        break
      end if
    end if
  end for
end function

Start:
  /* Standard UL/DL allocation */
  $n_{\text{UL}} \leftarrow 0$
  $n_{\text{DL}} \leftarrow 0$
  $N_T \leftarrow \lceil \frac{2 \cdot n}{n_{\text{RB}}} \rceil$
  if $N_T \,\%\, 2 == 0$ **then**
    for $t \in \{1, \ldots, N_T\}$ **do**
      ALLOCATE_UL_DL(t)
    end for
  else
    for $t \in \{1, \ldots, N_T - 2\}$ **do**
      ALLOCATE_UL_DL(t)
    end for

    /* allocate all remaining uplink resources */
    for $b \in \{1, \ldots, n \,\%\, n_{\text{RB}}\}$ **do**
      $n_{\text{UL}} \leftarrow n_{\text{UL}} + 1$
      allocate $\text{UL}_{n_{\text{UL}},b,N_T-1}$
    end for

    /* fill second-last T_{CPP} with downlink resources */
    for $b \in \{n \,\%\, n_{\text{RB}} + 1, \ldots, n_{\text{RB}}\}$ **do**
      $n_{\text{DL}} \leftarrow n_{\text{DL}} + 1$
      allocate $\text{DL}_{n_{\text{DL}},b,N_T-1}$
    end for

    /* allocate all remaining downlink resources */
    for $b \in \{1, \ldots, n_{\text{RB}}\}$ **do**
      $n_{\text{DL}} \leftarrow n_{\text{DL}} + 1$
      allocate $\text{DL}_{n_{\text{DL}},b,N_T}$
      if $n_{\text{DL}} == n$ **then**
        break
      end if
    end for
  end if

6.5 Performance Metrics

The overall system performance are assessed through the following steps:

1. **DL-based pipeline**. We define a cost model C (Sec. 6.5) to evaluate the learning performance of DL-based pipeline (Sec. 6.6).

2. **RTT analysis**. We perform a RTT analysis of the whole system (Sec. 6.5), with a particular focus on 5G network, and we evaluate the performance of 5G NR setups in Sec. 6.6.

3. **End-to-end performance analysis**. We assess the compatibility between the presented architectures and the DL-based pipeline while varying the number of AGVs (Sec. 6.6), also taking into account the execution time of the DL-based pipeline.

To perform these assessments, we need to define the performance metrics, which include the cost model C and RTT.

Cost Model

In binary classification settings, the last activation function of DL models is usually a sigmoid, which produces continuous values in the interval $[0, 1]$. Generally, 0.5 is the default threshold used to round the output values to either 0 or 1. However, this threshold can be tuned to minimize a cost model C through an iterative algorithm. In particular, the expression of C for a DL model X over a set of K time series $S = \{S_1, S_2, \ldots, S_K\}$ can be formulated as follows:

$$C = \sum_{k=1}^{K} \sum_{p=1}^{P_k} C_{\mathrm{FP}} + \sum_{k=1}^{K} \sum_{q=1}^{Q_k} C_{\mathrm{FN}}(s_q, S_k, m), \tag{6.5}$$

where P_k is the number of false positive samples for the k-th time series, Q_k is the number of false negative samples for the k-th time series, C_{FP} is the cost for a false positive sample, C_{FN} is the cost for a false negative sample. The expressions for C_{FP} and C_{FN} highly depend on the specific application requirements. For instance, in this case, in order to penalize more false negative samples rather than false positive samples, we set:

$$\begin{aligned} C_{\mathrm{FP}} &= 0.2 \\ C_{\mathrm{FN}}(s_q, S_k, m) &= m - |S_k| + q, \end{aligned} \tag{6.6}$$

where m is the margin, S_k is a time series, and q is the index of s_q in S_k. We set the cost of false positives C_{FP} constant, regardless of when they occur. On the other hand, the cost of false

negatives, represented by C_{FN}, escalates as the sample approaches the liquid fall. As a safety-critical application, the cost model focuses on false negatives (i.e., the missed anomalies) rather than false positives (i.e, the false alarms).
Nevertheless, the primary measure for evaluating the effectiveness of the DL-based pipeline is the average advance function $\bar{a}(D_X)$. Here, $D_X = \{s_1, s_2, \ldots, s_K\}$ represents the set of the initial samples in the time series correctly identified as faulty by a specific model X with a given margin m. This metric indicates the time duration before the actual fault occurrence and is defined as follows:

$$\bar{a}(D_X) = \frac{\sum_{i=1}^{K} a(s_i)}{K}, \tag{6.7}$$

where $a(s_i)$ is the advance function which indicates the amount of time before the actual fault occurs after sample s_i.

Round Trip Time analysis

In this work, the primary metric under evaluation is RTT. This metric represents the time elapsed between the generation of a data sample by an AGV and the subsequent execution of an action, after having received a command from the server. Let us analyze which are the contributions to RTT R:

$$R = T_{\mathrm{5G}} + T_{\mathrm{PS}} + T_A, \tag{6.8}$$

where:

- T_{5G} is the delay contribution introduced by 5G network;
- T_{PS} is the delay introduced by the DL-based pipeline for RUL estimation;
- T_A is the delay introduced by actuation performed at AGV side after the reception of a command.

T_{5G} can be decomposed as follows:

$$\begin{aligned} T_{5G} &= T_{\mathrm{P\_UE}} + T_{\mathrm{RAN\_UL}} + T_{\mathrm{P\_gNB}} + T_{\mathrm{CORE}} + T_{\mathrm{CORE}} + T_{\mathrm{P\_gNB}} + T_{\mathrm{RAN\_DL}} + T_{\mathrm{P\_UE}} = \\ &= 2 \cdot (T_{\mathrm{P\_UE}} + T_{\mathrm{P\_gNB}} + T_{\mathrm{CORE}}) + T_{\mathrm{RAN\_UL}} + T_{\mathrm{RAN\_DL}} \end{aligned} \tag{6.9}$$

This delay contribution, that is studied mainly using the simulator described in Sec. 6.4, includes the ones introduced by 5G NR and 5CN. The components are:

- T_{CORE} is the one-way delay introduced by the 5CN, and its value is provided by TIM, who conducted experiments on its own network. Due to signed NDA, it is not possible to explain how the results have been obtained.

- $T_{\text{P\_UE}}$ represents the time required by a UE to process data during both transmission and reception, as it traverses from the PHY layer to the application layer and vice versa;

- $T_{\text{P\_gNB}}$ represents the time required by a gNB to process data during both transmission and reception, as it traverses from the PHY layer to the network layer and vice versa;

- $T_{\text{RAN\_UL}}$ is the time needed to perform a successful RAN transmission in uplink;

- $T_{\text{RAN\_DL}}$ is the time needed to perform a successful RAN transmission in downlink;

We can introduce two additional terms:

$$T_{\text{NR}} = 2 \cdot (T_{\text{P\_UE}} + T_{\text{P\_gNB}}) + T_{\text{RAN\_UL}} + T_{\text{RAN\_DL}} \tag{6.10}$$

$$T_{\text{CN}} = 2 \cdot T_{\text{CORE}} \tag{6.11}$$

that represents the contribution of 5G NR and 5CN to RTT, respectively. By substituting (6.10) and (6.11) in (6.8), we obtain:

$$R = T_{\text{NR}} + T_{\text{CN}} + T_{\text{PS}} + T_A, \tag{6.12}$$

that highlights all the different contribution to RTT. We also introduce:

$$\overline{R} = \overline{T_{\text{NR}}} + \overline{T_{\text{CN}}} + \overline{T_{\text{PS}}} + \overline{T_A}, \tag{6.13}$$

that is (6.12) averaged on the total number of AGVs N and the total number of simulations N_S.

6.6 Numerical Results

In this section, we present our experimental setup and the numerical results achieved, with the aim of i) delving into the details of data-driven RUL prediction pipeline as described in Sec. 6.2, including the cost model C; ii) showing the impact of several parameters on 5G NR performance; and iii) illustrating the global performance of the system, taking into account both network and RUL prediction performance.

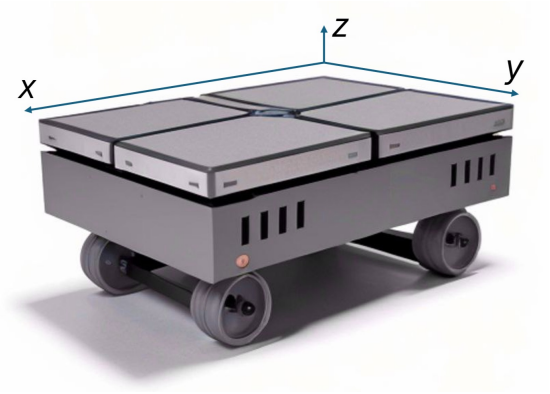


Figure 6.4: AGV's reference system (x, y, z) for A_x, A_y, and G_z.

Sensor Data Collection and Processing

We conducted an experimental campaign to collect real-time data from an AGV (see Fig. 6.4) navigating the industrial pilot line at BI-REX[2]. For each data sample, the following features were recorded:

- T_C: Collection timestamp;
- A_x: Axial acceleration along the primary axis x of the AGV, parallel to its main direction of movement;
- A_y: Axial acceleration along the secondary axis y, perpendicular to the main movement direction and the vertical axis;
- G_z: Angular acceleration along the vertical axis z, perpendicular to the factory floor;
- $P_{\hat{x}}$, $P_{\hat{y}}$: AGV position on the factory floor;
- O: AGV orientation relative to the factory floor;

Multiple navigation sessions were recorded, with each session involving the AGV carrying a water bottle. During these sessions, a sudden trajectory change caused the bottle to fall, allowing us to label the sensor data accurately. A custom script captured the timestamp of the fall, labeling the corresponding sensor data as a *Fault* event. All other data points were labeled as *Non-Fault*. The data collected during each session formed a time series, representing the AGV's movement over time, sampled at regular 100 ms intervals, yielding an ordered sequence of observations. All collected data have been made publicly available[3].
The collected real-time data underwent several pre-processing steps to address the following issues:

[2]BI-REX is an Italian Competence Center for Industry 4.0 (see https://bi-rex.it/).
[3]Dataset: https://www.kaggle.com/datasets/lorenzoamorosa/5g-industrial-iot-for-remaining-useful-life/.

- **Imbalanced data.** In RUL scenarios, the data are typically imbalanced due to the infrequency of failures relative to normal operations. This imbalance can bias DL algorithms toward favoring the majority class. To mitigate this, we applied class weighting [151], a common technique in RUL estimation tasks for handling imbalanced datasets.
- **Feature creation.** This technique can improve the performance of DL models by capturing more informative patterns for RUL estimation. We extracted statistical features such as mean, maximum, minimum, and standard deviation from fixed-length windows over the axial and angular acceleration data [152]. Additionally, we calculated derivatives by measuring variations between consecutive data points.
- **Stationarity.** Time series data often exhibit trends and seasonality, leading to non-stationary behavior [153]. We employed differencing techniques to enforce stationarity, thereby enhancing the precision of RUL estimations by revealing clearer patterns in the data. Specifically, we subtracted the average axial and angular acceleration values from each data point based on the AGV's position and orientation.
- **Standardization.** Finally, we standardized each feature to accelerate training convergence and ensure consistent scaling. This step prevents features with larger magnitudes from dominating the model, thereby improving overall performance.

RUL prediction pipeline performance

The RUL problem has been tackled using three distinct margins $m \in \{5, 10, 15\}$. Higher margins were also considered; however, they did not lead to good results. Empirically, it was found that a fall event is attributable on average to braking occurring from the immediately preceding second. Consequently, a too high margin is ineffective.
In Table 6.2 all the DL algorithms listed in Sec. 6.2 are assessed for the three considered margins. The data have been collected through two measurement campaigns, carried out on distinct days for the training and test sets. The data encompass 28 faults for the training set (corresponding to almost 33.500 training data points) and 13 faults for the test set (corresponding to almost 14.000 test data points).
In particular, Table 6.2 shows that a low margin (i.e., $m = 5$) corresponds to a low average advance and cost, while a high margin (i.e., $m = 10$ and $m = 15$) corresponds to a higher average advance and cost. This is a direct consequence of the fact that high margins increase both the maximum average advance achievable and learning task complexity, since more samples far from the actual liquid fall are labeled as faulty.
This performance trade-off poses constraints on network architectures employed in the IIoT system and the number of AGVs that can be served simultaneously. This analysis is performed in Sec. 6.6.

	$m = 5$		$m = 10$		$m = 15$	
Model X	C	$\bar{a}(D_{\mathrm{X}})$	C	$\bar{a}(D_{\mathrm{X}})$	C	$\bar{a}(D_{\mathrm{X}})$
BASELINE	80.80	0.24s	306.40	0.46s	448.20	1.07s
LR	96.00	0.32s	221.20	0.44s	679.60	0.45s
DNN	43.40	0.27s	142.00	0.66s	311.20	0.95s
1D-CNN	**28.80**	**0.27s**	**114.40**	**0.80s**	**197.60**	**1.33s**
AE	2396.40	0.39s	2666.80	0.90s	2561.80	1.40s
LSTM	95.40	0.20s	346.40	0.34s	569.80	0.81s
BiLSTM	61.60	0.28s	272.40	0.48s	689.40	1.08s
GRU	85.40	0.23s	290.80	0.68s	539.80	0.92s
VT	67.00	0.22s	316.80	0.47s	511.80	0.91s

Table 6.2: Cost and average advance function of seven DL models and a baseline threshold-based approach for three different margins.

Overall, 1D convolutional neural networks (1D-CNNs) are the best-performing models, as it can be seen from Table 6.2. They were the most effective models in capturing the local temporal patterns in the data related to the RUL prediction. The evidence shows also that complex memory-based models such as long-short term memory (LSTM), bi-directional long short term memory (BiLSTM), and gated recurrent unit (GRU) are not effective in this particular RUL estimation task. The reason for this is that liquid falls prediction mainly requires a small number of significant input samples, while recurrent models are specifically designed to capture long-term dependencies and patterns found in time series data. Another significant finding is that the reconstruction error, which autoencoders aim to minimize, might not serve as an effective predictor for the RUL. These models demonstrated notably lower performance compared to the alternative approaches. Despite their capabilities, vanilla transformers (VTs) fall short of reaching their full potential due to the limited availability of training data and a relatively low number of features compared to more prominent large language models (LLMs) [154]. Moreover, when applied to forecasting tasks involving time series data, transformers models encounter additional fundamental limitations, as discussed in [155].

5G NR performance

Using the simulator presented in Sec. 6.4, an analysis on 5G NR performance is conducted, with a deep focus on $\overline{T_{\mathrm{NR}}}$ evaluation, as defined in (6.13). Unless stated otherwise, the parameters used for the simulations are the ones reported in Table 6.3. We will often refer to two different operating frequency ranges: FR1 (with $f_c = 3.5$ GHz and $\Delta f = 30$ kHz) and FR2 (with $f_c = 28$ GHz and $\Delta f = 60$ kHz), where f_c represents the carrier frequency and Δf represents

[4] 32 byte is the payload used in experiments conducted in Sec. 6.6. We considered also a payload of 500 byte, assuming that other data could be used for RUL estimation.

Table 6.3: Simulation parameters

Parameter	Description	Value
B	Total system bandwidth	$\{25, 50\}$ MHz
Δf	Subcarrier spacing	$\{30, 60\}$ kHz
f_c	Carrier frequency	$\{3.5, 28\}$ GHz
N_G	Number of gNBs	$\{1, 2\}$
P_{UL}	Uplink payload	$\{32, 500\}$ byte[4]
P_{DL}	Downlink payload	1 byte
μ_{DL}	Mean of downlink probability distribution	300 ms
σ_{DL}	Standard deviation of downlink probability distribution	100 ms
ξ	Maximum imbalance factor	0
$T_{\mathrm{P\_gNB}}$	gNB processing time	7 OFDM symbols
$T_{\mathrm{P\_UE}}$	UE processing time	7 OFDM symbols
T_{CPP}	Control Plane periodicity	8 mini-slots
T_{S}	Simulation time	10 s
τ_{UL}	Uplink periodicity	100 ms
H	5G protocol stack header	72 byte
N_{S}	Number of simulations	10
l	Factory plant length	1000 m
w	Factory plant width	150 m
h	Factory plant height	10 m
γ	Plot confidence interval	99%
T	Noise temperature	290 K
$P_{\mathrm{TX\_UE}}$	UE transmit power	23 dBm
$P_{\mathrm{TX\_gNB}}$	gNB transmit power	30 dBm
G_{UE}	UE antenna gain	0 dB
G_{gNB}	gNB antenna gain	0 dB
Q_{DL}	Maximum allowed downlink latency	25 ms
Q_{UL}	Maximum allowed uplink latency	75 ms

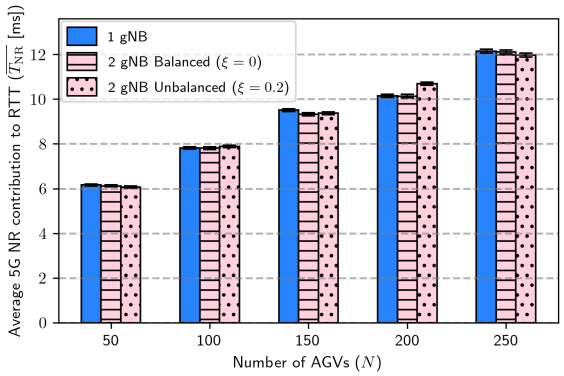


Figure 6.5: $\overline{T_{\text{NR}}}$ as a function of N, N_G and ξ, considering FR1 and $B = 25$ MHz.

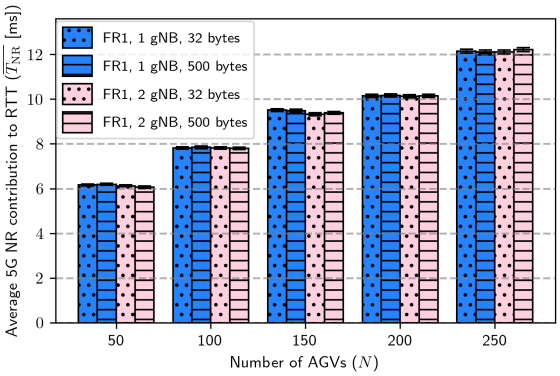


Figure 6.6: $\overline{T_{\text{NR}}}$ as a function of N, N_G and P_{UL}, considering FR1 and $B = 25$ MHz.

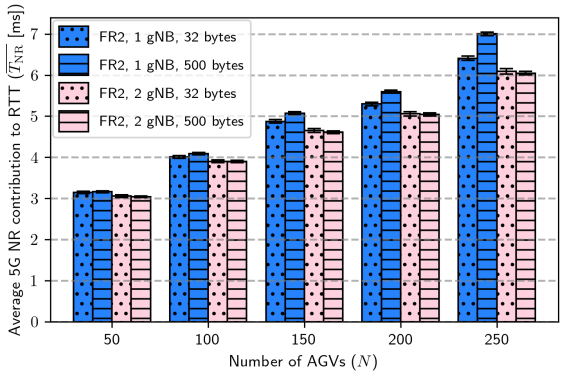


Figure 6.7: $\overline{T_{\text{NR}}}$ as a function of N, N_G and P_{UL}, considering FR2 and $B = 50$ MHz.

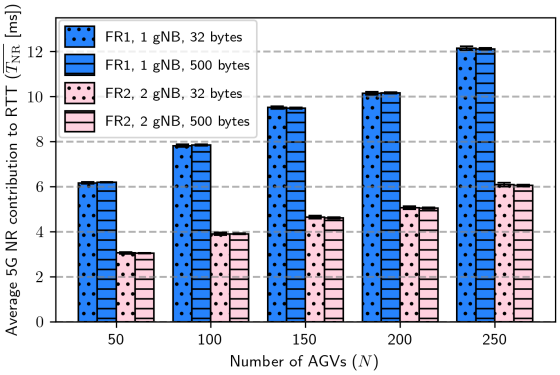


Figure 6.8: $\overline{T_{\text{NR}}}$ as a function of N, P_{UL} and considered frequency range (FR1, FR2). $B = 25$ MHz in FR1 and $B = 50$ MHz in FR2.

the SCS.

Impact of the clustering algorithm. We first study the effect of clustering algorithm on $\overline{T_{\text{NR}}}$, focusing on the impact of the imbalance factor ξ.

As shown in Fig. 6.5, different scenarios yield very similar results in terms of $\overline{T_{\text{NR}}}$, irrespective from the number of gNBs N_G and ξ. It is noteworthy that with 2 gNBs, the performance is more similar to the case with 1 gNB when we have balanced clusters. This is due to the fact that, in case of imbalanced clusters, we might have a higher number of AGVs associated to a gNB that leads to requiring an higher number of T_{CPP} in a T_{SRP}. Imbalanced clusters are typically employed when we have heterogeneous gNBs or to perform load balancing. Given that our current scenario does not fall into either of these categories, we will proceed with $\xi = 0$, indicating balanced clusters for the subsequent tests.

Impact of a multi-gNB system. The second experiment aims to understand when 2 gNBs are beneficial in our scenario. In Fig. 6.6, we can see the impact of having multiple gNBs in FR1

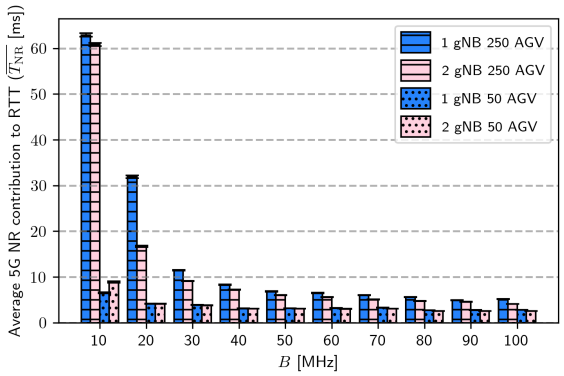


Figure 6.9: $\overline{T_{\text{NR}}}$ as a function of N, N_G and B, considering FR2 and $P_{\text{UL}} = 500$ byte.

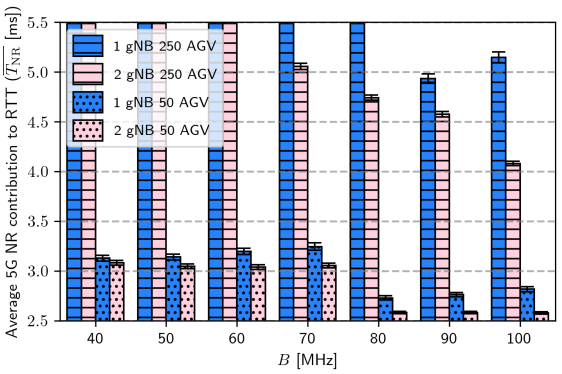


Figure 6.10: $\overline{T_{\text{NR}}}$ as a function of N, N_G and B, considering FR2, $P_{\text{UL}} = 500$ byte, $B \in \{40, 50, 60, 70, 80, 90, 100\}$ MHz, and $\overline{T_{\text{NR}}} \in [2.5, 5.5]$ ms.

with different payloads P_{UL}. The performance in terms of $\overline{T_{\text{NR}}}$ is nearly identical. Since there is no improvement going from 1 to 2 gNBs, this implies that the SNR improvement provided by 2 gNBs in FR1 is negligible. Moreover, since $\overline{T_{\text{NR}}}$ does not change with P_{UL}, this suggests that the control plane, rather than the data plane, is the bottleneck of the system. We conclude that 2 gNBs are useless in FR1.

In Fig. 6.7, we can see the impact of having multiple gNBs in FR2 with different payloads P_{UL}. Unlike the previous case, now performance is better when using 2 gNBs. In particular, the larger the payload P_{UL} and the greater the number of AGVs N, the more pronounced the improvement introduced by 2 gNBs. This outcome suggests that data plane is limiting performance when using 1 gNB. When using 2 gNBs, performance does not depend on the payload size P_{UL}: this suggests that, thanks to the higher average SNR, that leads to a lower number of re-transmissions and an higher spectral efficiency, the data plane is no longer the bottleneck of the system, but the control plane is. We conclude that 2 gNBs are useful in FR2.

The experiments reveal that doubling the number of gNBs does not uniformly improve system performance, it depends on which system component acts as the bottleneck. In FR1, the performance metric $\overline{T_{\text{NR}}}$ remains nearly identical whether using one or two gNBs. This is because the SNR in FR1 is already sufficient, and the control plane is the limiting factor. When two gNBs are deployed, the available constant bandwidth is simply split between them, meaning that each gNB ends up with fewer control-plane resources. Since the control-plane capacity governs the system's performance in FR1, the potential SNR gains from an extra gNB do not translate into measurable improvement. Conversely, in FR2 the scenario is different. Here, especially under conditions of larger payloads and a higher number of AGVs, the performance improvement with two gNBs becomes evident. In FR2 the system can be data-plane limited, meaning that factors like SNR, retransmissions, and spectral efficiency play a significant role. The use of

two gNBs in FR2 boosts the average SNR, which in turn reduces retransmissions and increases data-plane efficiency. As a result, the performance improves when data transmission is the main challenge. However, it is important to note that once the data plane is no longer the bottleneck, the same constant control-plane resource (divided equally between the two gNBs) again limits the overall performance. This is why for small payloads, where the control plane dominates, the improvement is marginal.

Impact of the operating frequency. In Fig. 6.8, we compare the performance of a system operating in FR1 and FR2. In FR1, we use 1 gNB, while in FR2, we employ 2 gNBs, based on previous research findings. To ensure a fair comparison, both configurations have an equal number of RBs, resulting in the same number of AGVs served during each T_{CPP}. It is possible to appreciate that, independently from P_{UL}, $\overline{T_{\text{NR}}}$ is halved in FR2 with respect to FR1. Since the bottleneck in both configurations lies in the control plane, particularly in the number of T_{CPP} within a T_{SRP}, one of the main advantages of operating in FR2 becomes apparent: a reduced T_{CPP} duration due to a shorter duration of the OFDM symbol. Specifically, in FR1, we have $\Delta f = 30$ kHz, while in FR2, we have $\Delta f = 60$ kHz. Consequently, in the latter case, the T_{CPP} duration is halved compared to the former. This justifies why $\overline{T_{\text{NR}}}$ is approximately halved in FR2 with respect to FR1.

Impact of the bandwidth. In Fig. 6.9, we study the impact of B in FR2 on the system. We consider $P_{\text{UL}} = 500$ byte, and we test the system with 1 and 2 gNBs, considering $N = 50$ and $N = 250$. As we can see, with $N = 250$ the general trend is that $\overline{T_{\text{NR}}}$ decreases with higher B, since there are more available RBs per AGV. With $N = 50$, there is no significant improvement by using 2 gNBs, and when $B = 10$ MHz, it is even detrimental. This is due to the fact that, in this extreme case, the overall amount of resources is so scarce that the improvement given by the higher SNR when using two gNBs is lower than that the disadvantage introduced by splitting resources.

In Fig. 6.10 we present the same data of Fig. 6.9, but focusing on B between 40 MHz and 100 MHz, in order to appreciate the step-wise performance improvement due to control plane. If the increase of B leads to an increase of the number of RBs sufficient to save one T_{CPP} when serving all the AGVs, then $\overline{T_{\text{NR}}}$ decreases sharply, as we can see in Fig. 6.10 where we have 2 gNBs, 50 AGVs, and we go from $B = 70$ MHz to $B = 80$ MHz. If the increase of B does not lead to a reduction of the number of T_{CPP} in a T_{SRP}, the only effect introduced is the increase of noise in the system, leading to worse performance. This effect is present only if data plane is not the bottleneck of the system and with 1 gNB, since we are in an SNR limited system.

Performance of the End-to-End RUL Chain

Given the definition of average RTT $\overline{R}$ (6.13), we consider:

- $\overline{T_{\text{NR}}}$ as studied in Sec. 6.6;

- $\overline{T_{\text{CN}}}$ as provided by TIM:
  - Architecture 1: $\overline{T_{\text{CN}}} = 14$ ms;
  - Architecture 2 and 3: $\overline{T_{\text{CN}}} = 4$ ms;
  - Architecture 4: $\overline{T_{\text{CN}}} = 2$ ms;
- $\overline{T_{\text{PS}}}$ is obtained through experimental tests using the best performing DL-based pipeline. The computing platform consisted in an i9-11900K processor with 128 GB of RAM, featuring 8 cores and 16 threads exploited through parallel programming. Despite we tested a single CPU, we assumed to have more CPUs, to perform load balancing between them, and that performance scales linearly with the number of CPUs.
- $\overline{T_A} = 200$ ms, derived from a commercial product[5].

In Fig. 6.11, we present the overall results comprising the performance of DL-based RUL prediction pipeline, 5G NR, and 5CN. We depict average RTT $\overline{R}$ as a function of N and different architectures employed, and we show the average advance provided by 1D-CNN $\bar{a}(D_{\text{1D-CNN}})$ with $m = 5$ and $m = 10$. It is noteworthy that the 5G network has the smallest contribution on the total average RTT $\bar{R}$: processing time and actuation time are significantly larger. It is then possible to appreciate the conditions under which we are able to prevent the failure, wherein the average RTT $\overline{R}$ of the system is lower than the advance $\bar{a}$. Despite the average RTT $\overline{R}$ being constantly lower than the average advance provided with $m = 10$, it is sub-optimal: in the case of $m = 5$, the cost C is lower, leading to better performance in terms of false positives and false negatives, as shown in Sec. 6.6. This implies that, when possible, it is preferable to choose to use $m = 5$. When the number of AGVs $N \leq 100$, average RTT $\overline{R}$ is lower than $\bar{a}(D_{\text{1D-CNN}})$ with $m = 5$, irrespective from the architecture. Conversely, for $N \geq 200$, average RTT $\overline{R}$ is greater than $\bar{a}(D_{\text{1D-CNN}})$ with $m = 5$, regardless of the architecture, leading to the use of $m = 10$. When $N = 150$, the architecture plays a fundamental role, determining which value of m to use: this is the only scenario in which having a NPN leads to a significant advantage. Finally, we note that although a margin of $m = 15$ allows for a greater average advance, it is excluded from the comparison due to its high cost and because the average advance obtained with $m = 10$ is always sufficient.

[5]Reference: https://www.hitbotrobot.com/product/z-efg-12-robotic-gripper/

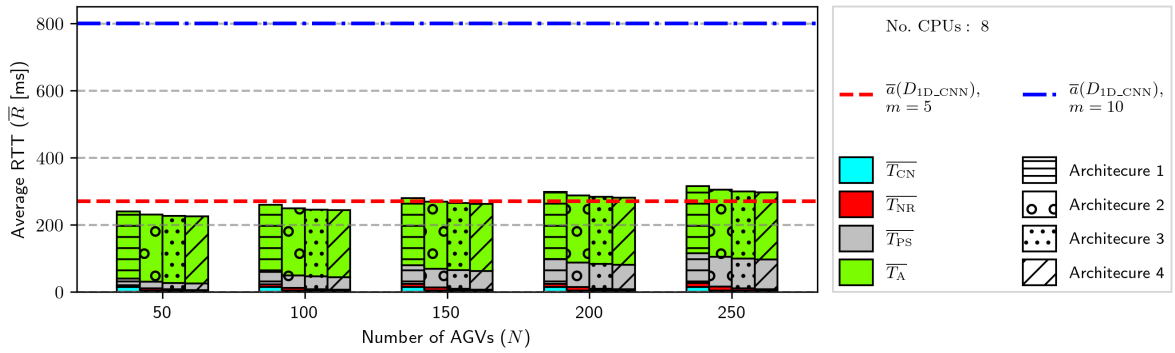


Figure 6.11: RTT $\overline{R}$ as a function of N and network architectures. The advances of the best performing RUL-based pipelines are represented with horizontal dashed lines. We assume to have 8 CPUs performing inference.

Chapter 7

Conclusions

This thesis has explored the confluence of wireless communications and artificial intelligence, with the overarching objective of facilitating the development of next-generation autonomous network systems. In pursuit of this objective, the work has yielded both technical innovations and significant insights into the challenges and benefits inherent in interdisciplinary research.
A principal conclusion drawn from this research is that the effective integration of artificial intelligence and wireless communications is fundamentally interdisciplinary, requiring a profound understanding of both domains. The research process encompassed several key aspects:

- The technical contributions presented, including the deep reinforcement learning (DRL)-based adaptive optimization framework (Chapter 3), the Bayesian generative models for synthetic data generation (Chapter 4), and the distributed learning approaches employing multi-agent reinforcement learning (MARL) and graph neural networks (GNNs) (Chapter 5), demonstrate that the successful fusion of theoretical AI concepts with the practical constraints of real-world wireless networks is both a complex undertaking and a critical necessity.

- While simulations offered a controlled environment conducive to the evaluation of novel algorithms, this work also illuminated the persistent discrepancies between simulated models and the complexities inherent in real-world deployments. Unpredictable interference, hardware imperfections, and dynamic mobility patterns, among other real-world phenomena, necessitate the development of robust and adaptable solutions. This was particularly evident in the 5G industrial internet of yhings (IIoT) case study and in the capacity and coverage optimization (CCO) problem leveraging real-world data.

- In applications where safety is paramount, the requirement for explainable and trustworthy artificial intelligence is of utmost importance. The Bayesian generative models developed in this thesis not only provide theoretical guarantees via uncertainty quantifi-

cation but also establish a precedent for the construction of AI systems that engender trust among decision-makers.

- The availability of high-quality, labeled data remains a significant limiting factor in numerous wireless scenarios. This thesis addressed data scarcity through the application of techniques such as synthetic data generation and distributed learning, thereby emphasizing that innovative approaches to data management are of equal importance to algorithmic advancements.

- For the successful convergence of artificial intelligence and wireless communications, the establishment of standards that accommodate both application-specific requirements and implementation details is essential. Such standardization would promote wider adoption and facilitate interoperability across a diverse range of systems.

For researchers embarking on studies in this area, these reflections may serve as guidance: embracing the inherent learning curve by extending beyond one's disciplinary specialization is crucial. A robust foundation in both artificial intelligence and wireless communications, coupled with a dedication to bridging theoretical progress with practical realities, will prove invaluable in navigating this intricate and rapidly evolving field.
While the primary focus of this thesis has been the development of robust AI-driven frameworks for wireless networks, the techniques introduced inherently facilitate a transition towards goal-oriented communications and learning—a promising avenue for future investigation.

- The DRL framework detailed in Chapter 3 is directly extensible to scenarios where reward functions are tailored to specific operational objectives. For instance, rather than optimizing for general performance metrics such as throughput, reward functions can be designed to prioritize objectives such as minimizing energy consumption or ensuring the timely delivery of critical control signals.

- The Bayesian generative model presented in Chapter 4 can be adapted to generate synthetic data that emphasizes rare or critical events, such as fault scenarios in industrial systems. This concept of goal-oriented data augmentation would be particularly beneficial for training systems in situations where such events are sparsely represented in real-world datasets.

- The distributed learning techniques introduced in Chapter 5 are inherently well-suited for multi-agent scenarios where collaboration is essential. By leveraging graph structures and GNNs, future research can explore cooperative frameworks in which agents communicate and coordinate their actions to achieve specific, shared objectives—for example, the rapid mapping of an unknown environment in a multi-robot exploration task.

– The 5G Industrial IoT case study presented in Chapter 6 exemplifies a scenario where all design choices are dictated by a clearly defined end goal—the estimation of the remaining useful life (RUL) of an Automated Guided Vehicle (AGV). This emphasis on a concrete objective illustrates how goal-oriented design can guide the integration of communication infrastructures and deep learning techniques to fulfill specific operational requirements.

By redefining the objectives of wireless networks to align with specific application goals, rather than generic performance metrics, future research can stimulate the development of more efficient, robust, and trustworthy systems, particularly in mission-critical and resource-constrained environments.

In conclusion, this thesis has advanced the state-of-the-art in AI-driven wireless networks while concurrently providing insights into the broader challenges associated with interdisciplinary research. By addressing critical issues such as the simulation-to-reality gap, the imperative for explainability, data scarcity, and the significance of standardization, this work establishes a robust foundation for future investigations into fully autonomous, goal-oriented network systems. It is the author's hope that these reflections, in conjunction with the technical contributions detailed throughout this thesis, will serve as inspiration and guidance for future researchers in their pursuit of innovative solutions at the dynamic intersection of artificial intelligence and wireless communications.

Appendix A

Appendices

A.1 Uncertainty analysis and Gaussian assumption for RSRP estimation

Here, we analyze the estimation uncertainty related to the Reference Signal Received Power (RSRP), which serves as an estimator for the median component of the received power $\mathcal{P}(\mathbf{s})$ at a fixed location $\mathbf{s} = (s_1, s_2)$. Since we are considering the role of the RSRP as an estimator of the median component $Me[\mathcal{P}(s_1, s_2)]$, we hereafter denote the estimator as $\hat{\mu}$, and the target value $Me[\mathcal{P}(s_1, s_2)]$ as μ. Our focus is to model the aleatoric uncertainty of $\hat{\mu}$ exclusively. This is crucial for establishing a fair prior assumption in a neural-probabilistic model for the task of RSRP regression. The RSRP, as defined by 3GPP [62], is a linear average over the narrowband instantaneous power of every resource element (RE) that carries cell-specific reference signals (CS-RS). When considering an Orthogonal Frequency Division Multiplexing (OFDM) scheme and a transmitter (eNB) equipped with channel equalization capability, the narrowband instantaneous power of a resource element (RE) can be assumed to be an independent and identically distributed (i.i.d.) sample of $\mathcal{P}(s_1, s_2)$. Furthermore, assuming proper countermeasures to small-scale fading, $\mathcal{P}(s_1, s_2)$ is mainly characterized by the shadowing noise σ_S, which follows a log-normal distribution. Hence, the median component of the received power in dB, $Me[\mathcal{P}(s_1, s_2)]$ is equivalent to the mean $\mathbb{E}[\mathcal{P}(s_1, s_2)]$. Accordingly, we can define the probability distribution of $\hat{\mu}$, $f_{\hat{\mu}}(\mu, \sigma_S)$, as the joint probability distribution of N i.i.d. realizations of $\mathcal{P}(s_1, s_2)$:

$$
\begin{aligned}
f_{\hat{\mu}}(\mu, \sigma_S) &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_S^2}} \exp \frac{(p_i - \mu)^2}{2\sigma_S^2} = \\
&= \left(\frac{1}{\sqrt{2\pi\sigma_S^2}} \right)^N \exp \frac{\sum_{i=1}^{N} (p_i - \mu)^2}{2\sigma_S^2} \,,
\end{aligned}
$$

where $p_i \sim \mathcal{P}(s_1, s_2)$ indicates the instantaneous i.i.d. received power sample. Since σ_S typically can be statistically determined, we can characterize the estimation uncertainty of the estimator $\hat{\mu}$, namely ε_μ, by means of its Cramér-Rao Lower Bound (CRLB):

$$\begin{aligned}
\text{CRLB}(\hat{\mu}) &= -\mathbb{E}\left[\frac{\partial^2 \ln f_{\hat{\mu}}(\mu, \sigma_S)}{\partial \mu^2}\right]^{-1} \\
&= -\mathbb{E}\left[\frac{\partial^2}{\partial \mu^2}\left(\ln\left(\frac{1}{\sqrt{2\pi\sigma_S^2}}\right)^N + \ln\frac{1}{2\sigma_S^2}\sum_{i=1}^{N}(p_i-\mu)^2\right)\right]^{-1} \\
&= \mathbb{E}\left[\frac{\partial}{\partial \mu}\left(\frac{1}{\sigma_S^2}\sum_{i=1}^{N}(\mu - p_i)\right)\right]^{-1} \\
&= \mathbb{E}\left[\frac{N}{\sigma_S^2}\right]^{-1} = \frac{\sigma_S^2}{N} = \varepsilon_\mu \, .
\end{aligned}$$

We derived an intuitive result from the equation above: the estimation uncertainty of $\hat{\mu}$ becomes negligible as $N \to \infty$. Given that N is proportional to the number of REs carrying CS-RS, which depends on the system bandwidth B, this indicates that the UE-measured RSRP is a Minimum Variance Unbiased (MVU) estimator, as the system bandwidth $B \to \infty$. Nevertheless, this formulation implies that ε_μ is the only source of error in the estimation process. When moving to a crowdsourcing setting, this assumption falls, as the total aleatoric uncertainty, ε_{al}, is characterized by additional error sources related to the diversity in receivers' equipment and measurement conditions. Leveraging on the central limit theorem, under the assumption of i.i.d. error sources, we can express the sum of these additional error sources as a new random variable $\varepsilon_M \sim \mathcal{N}(0, \sigma_M^2)$. Thus, the aleatoric uncertainty ε_{al} of $\hat{\mu}$ can be described as the sum of a Normal random variable ε_μ with deterministic and reducible variance σ_S^2/N, and a second random variable ε_M, with irreducible and unknown variance σ_M^2. Given that the sum of two independent Normal random variables is a Normal random variable, the following holds:

$$\varepsilon_{al} = \varepsilon_\mu + \varepsilon_M \Rightarrow \varepsilon_{al} \sim \mathcal{N}(0, \sigma_S^2/N + \sigma_M^2) \, .$$

This provides an analytical interpretation behind the assumption of a Gaussian prior over the predicted RSRP, for any given fixed location (s_1, s_2). In Sec. 4.6, the validity of our analytical derivations is further ascertained by means of a calibration analysis, which shows that the proposed neural-probabilistic regression is not affected by model miss-specification.

List of Publications

The research activities described in this thesis have produced the following set of contributions:

1. Longhi, N., Amorosa, L. M., Cavallero, S., Buracchini, E., Verdone, R., "5G Architectures Enabling Remaining Useful Life Estimation for Industrial IoT: a Holistic Study", IEEE Open Journal of the Communications Society, 2025.

2. Skocaj, M., Amorosa, L. M., Lombardi, M., Verdone, R. "GUMBLE: Uncertainty-Aware Conditional Mobile Data Generation using Bayesian Learning", IEEE Transactions on Mobile Computing, 2024.

3. Amorosa, L. M., Skocaj, M., Verdone, R., Gündüz, D. "Multi-Agent Reinforcement Learning for Power Control in Wireless Networks via Adaptive Graphs", IEEE ICC, 2024.

4. Amorosa, L. M., Longhi, N., Cuozzo, G., Bachan, W., Lieti, V., Buracchini, E., Verdone, R., "An End-To-End Analysis of Deep Learning-Based Remaining Useful Life Algorithms for Satefy-Critical 5G-Enabled IIoT Networks", IEEE PIMRC, 2023.

5. Skocaj, M., Amorosa, L. M., Ghinamo, G., Muratore, G., Micheli, D., Zabini, F., Verdone, R. "Optimization of the configuration of a Mobile Communications Network", World Patent WO2023156301A1, 2023.

6. Skocaj, M., Amorosa, L. M., Ghinamo, G., Muratore, G., Micheli, D., Zabini, F., Verdone, R. "Cellular network capacity and coverage enhancement with MDT data and deep reinforcement learning", Computer Communications, 2022.

The following set of contributions are not included in this thesis.

7. Amorosa, L. M., Gao, Z., Verdone, R., Popovski, P., Gündüz, D., "Learning to Send Shared Messages Using a Decentralized Medium Access Control Protocol", *to be submitted to* IEEE Transactions on Communications.

8. Amorosa, L. M., Chahoud, T., Gao, Z., Verdone, R., Gündüz, D., "GNN-based Power Allocation for Industrial IoT Networks with Varying Density", IEEE ICMLCN, 2025.

9. Giovannini, A., Campolo, C., Todisco, V., Molinaro, A., Amorosa, L. M., Lei, L., Bazzi, A., "Path Selection Based on Network Service Quality for Infrastructure-Assisted Automated Driving", IEEE WCNC, 2025.

Bibliography

[1] Osvaldo Simeone. *Machine learning for engineers*. Cambridge university press, 2022.

[2] Yonina C Eldar, Andrea Goldsmith, Deniz Gündüz, and H Vincent Poor. *Machine learning and wireless communications*. Cambridge University Press, 2022.

[3] Fa-Long Luo. *Machine learning for future wireless communications*. John Wiley & Sons, 2020.

[4] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". In: *IEEE Communications Surveys & Tutorials* 21.4 (2019), pp. 3133–3174.

[5] Tugba Erpek, Timothy J O'Shea, Yalin E Sagduyu, Yi Shi, and T Charles Clancy. "Deep learning for wireless communications". In: *Development and Analysis of Deep Learning Architectures* (2020), pp. 223–266.

[6] Deniz Gündüz, Paul de Kerret, Nicholas D. Sidiropoulos, David Gesbert, Chandra R. Murthy, and Mihaela van der Schaar. "Machine Learning in the Air". In: *IEEE Journal on Selected Areas in Communications* 37.10 (2019), pp. 2184–2199.

[7] Jithin Jagannath, Nicholas Polosky, Anu Jagannath, Francesco Restuccia, and Tommaso Melodia. "Machine learning for wireless communications in the Internet of Things: A comprehensive survey". In: *Ad Hoc Networks* 93 (2019), p. 101913.

[8] Manuel Eugenio Morocho-Cayamcela, Haeyoung Lee, and Wansu Lim. "Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions". In: *IEEE Access* 7 (2019), pp. 137184–137206.

[9] Osvaldo Simeone. "A Very Brief Introduction to Machine Learning With Applications to Communication Systems". In: *IEEE Transactions on Cognitive Communications and Networking* 4.4 (2018), pp. 648–664.

[10] Xingqin Lin. "Artificial Intelligence in 3GPP 5G-Advanced: A Survey". In: *arXiv preprint arXiv:2305.05092* (2023).

[11] 3GPP. *Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR air interface*. Tech. rep. 2023.

[12] Henrik Hellström, José Mairton B da Silva Jr, Mohammad Mohammadi Amiri, Mingzhe Chen, Viktoria Fodor, H Vincent Poor, Carlo Fischione, et al. "Wireless for Machine Learning: A Survey". In: *FTSCP* 15.4 (2022), pp. 290–399.

[13] Christo Kurisummoottil Thomas, Christina Chaccour, Walid Saad, Merouane Debbah, and Choong Seon Hong. "Causal Reasoning: Charting a Revolutionary Course for Next-Generation AI-Native Wireless Networks". In: *arXiv preprint arXiv:2309.13223* (2023).

[14] Yong Xiao, Guangming Shi, Yingyu Li, Walid Saad, and H. Vincent Poor. "Toward Self-Learning Edge Intelligence in 6G". In: *IEEE Communications Magazine* 58.12 (2020), pp. 34–40.

[15] Walid Saad, Mehdi Bennis, and Mingzhe Chen. "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems". In: *IEEE Network* 34.3 (2020), pp. 134–142.

[16] Jakob Hoydis, Fayçal Ait Aoudia, Alvaro Valcarce, and Harish Viswanathan. "Toward a 6G AI-Native Air Interface". In: *IEEE Communications Magazine* 59.5 (2021), pp. 76–81.

[17] Lauri Lovén, Teemu Leppänen, Ella Peltonen, Juha Partala, Erkki Harjula, Pawani Porambage, Mika Ylianttila, and Jukka Riekki. "EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks". In: *6G Wireless Summit, March 24-26, 2019 Levi, Finland* (2019).

[18] Wen Wu, Conghao Zhou, Mushu Li, Huaqing Wu, Haibo Zhou, Ning Zhang, Xuemin Sherman Shen, and Weihua Zhuang. "AI-Native Network Slicing for 6G Networks". In: *IEEE Wireless Communications* 29.1 (2022), pp. 96–103.

[19] Khaled B Letaief, Yuanming Shi, Jianmin Lu, and Jianhua Lu. "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications". In: *IEEE Journal on Selected Areas in Communications* 40.1 (2021), pp. 5–36.

[20] Tze-Yang Tung, Szymon Kobus, Joan Pujol Roig, and Deniz Gündüz. "Effective Communications: A Joint Learning and Communication Framework for Multi-Agent Reinforcement Learning Over Noisy Channels". In: *IEEE Journal on Selected Areas in Communications* 39.8 (2021), pp. 2590–2603.

[21] Emilio Calvanese Strinati and Sergio Barbarossa. "6G networks: Beyond Shannon towards semantic and goal-oriented communications". In: *Computer Networks* 190 (2021), p. 107930.

[22] Luciano Miuccio, Salvatore Riolo, Sumudu Samarakoon, Daniela Panno, and Mehdi Bennis. “Learning Generalized Wireless MAC Communication Protocols via Abstraction”. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 2322–2327.

[23] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. “Learning to communicate with deep multi-agent reinforcement learning”. In: *Advances in neural information processing systems* 29 (2016).

[24] Mateus P Mota, Alvaro Valcarce, Jean-Marie Gorce, and Jakob Hoydis. “The emergence of wireless MAC protocols with multi-agent reinforcement learning”. In: *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2021, pp. 1–6.

[25] Sejin Seo, Jihong Park, Seung-Woo Ko, Jinho Choi, Mehdi Bennis, and Seong-Lyun Kim. “Toward Semantic Communication Protocols: A Probabilistic Logic Perspective”. In: *IEEE Journal on Selected Areas in Communications* 41.8 (2023), pp. 2670–2686.

[26] 3rd Generation Partnership Project (3GPP). *3GPP TR 36.902*. Technical Report 36.902. 3GPP, 2009.

[27] 3rd Generation Partnership Project (3GPP). *3GPP TS 36.300*. Technical Specification 36.300. 3GPP, 2009.

[28] 3rd Generation Partnership Project (3GPP). *3GPP TS 32.500*. Technical Specification 32.500. 3GPP, 2022.

[29] European Telecommunications Standards Institute (ETSI). *ETSI GS ZSM 001 V1.1.1 (2019-10): Zero-touch network and Service Management (ZSM); Requirements based on documented scenarios*. Technical Report GS ZSM 001 V1.1.1. ETSI, Oct. 2019.

[30] 3rd Generation Partnership Project (3GPP). *TS 28.312: Management and orchestration; Intent-driven management services for mobile networks*. Technical Specification. 3GPP, 2023.

[31] 3rd Generation Partnership Project (3GPP). *TR 28.912: Study on enhanced intent-driven management services for mobile networks*. Technical Report. 3GPP, 2023.

[32] European Telecommunications Standards Institute (ETSI). *ETSI GR ZSM 011 V1.1.1 (2023-02): Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects*. Technical Report GR ZSM 011 V1.1.1. ETSI, Feb. 2023.

[33] Matteo Zecchin, Sangwoo Park, Osvaldo Simeone, Marios Kountouris, and David Gesbert. “Robust bayesian learning for reliable wireless ai: Framework and applications”. In: *IEEE Transactions on Cognitive Communications and Networking* (2023).

[34] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. "Multi-agent reinforcement learning: A selective overview of theories and algorithms". In: *Handbook of reinforcement learning and control* (2021), pp. 321–384.

[35] Sven Gronauer and Klaus Diepold. "Multi-agent deep reinforcement learning: a survey". In: *Artificial Intelligence Review* (2022), pp. 1–49.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[37] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[38] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[39] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight uncertainty in neural network". In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[40] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[41] Oliver Dürr, Beate Sick, and Elvis Murina. *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications, 2020.

[42] William L Hamilton, Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications". In: *arXiv preprint arXiv:1709.05584* (2017).

[43] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[44] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[45] Viktor Berggren, Rafia Inam, Leonid Mokrushin, Alberto Hata, Jaeseong Jeong, Swarup Kumar Mohalik, Julien Forgeat, and Stefano Sorrentino. *Artificial Intelligence in next generation connected systems*. Tech. rep. Sept. 2021.

[46] Osianoh Glenn Aliu, Ali Imran, Muhammad Ali Imran, and Barry Evans. "A Survey of Self Organisation in Future Cellular Networks". In: *IEEE Communications Surveys Tutorials* 15.1 (2013), pp. 336–361.

[47] 3rd Generation Partnership Project (3GPP). *TS 37.320 - Universal Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collection for Minimization of Drive Tests (MDT); Overall description; Stage 2*. Rel 16.6.0. Sept. 2021.

[48] Wuri A. Hapsari, Anil Umesh, Mikio Iwamura, Malgorzata Tomala, Bodog Gyula, and Benoist Sebire. "Minimization of drive tests solution in 3GPP". In: *IEEE Communications Magazine* 50.6 (2012), pp. 28–36.

[49] Chiara Mizzi, Alessandro Fabbri, Sandro Rambaldi, Flavio Bertini, Nico Curti, Stefano Sinigardi, Rachele Luzi, Giulia Venturi, D. Micheli, Giuliano Muratore, Aldo Vannelli, and Armando Bazzani. "Unraveling pedestrian mobility on a road network using ICTs data during great tourist events". In: *EPJ Data Science* 7 (Dec. 2018).

[50] Andrea Scaloni, Pasquale Cirella, Mauro Sgheiz, Riccardo Diamanti, and Davide Micheli. "Multipath and Doppler Characterization of an Electromagnetic Environment by Massive MDT Measurements From 3G and 4G Mobile Terminals". In: *IEEE Access* 7 (2019), pp. 13024–13034.

[51] Davide Micheli and Giuliano Muratore. "Smartphones Reference Signal Received Power MDT Radio Measurement Statistical Analysis Reveals People Feelings during Music Events". In: *2019 PhotonIcs Electromagnetics Research Symposium - Spring (PIERS-Spring)*. 2019, pp. 427–437.

[52] *TS 36.902 - Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions*. Rel 9.3.1. 3GPP. Apr. 2011.

[53] *TS 32.500 - Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements*. Rel 16.0.0. 3GPP. July 2020.

[54] Osvaldo Simeone. "A Very Brief Introduction to Machine Learning With Applications to Communication Systems". In: *IEEE Transactions on Cognitive Communications and Networking* 4.4 (2018), pp. 648–664.

[55] Mirza Golam Kibria, Kien Nguyen, Gabriel Porto Villardi, Ou Zhao, Kentaro Ishizu, and Fumihide Kojima. "Big Data Analytics, Machine Learning, and Artificial Intelligence in Next-Generation Wireless Networks". In: *IEEE Access* 6 (2018), pp. 32328–32338.

[56] Sascha Berger, Albrecht Fehske, Paolo Zanier, Ingo Viering, and Gerhard Fettweis. "Online Antenna Tilt-Based Capacity and Coverage Optimization". In: *IEEE Wireless Communications Letters* 3.4 (2014), pp. 437–440.

[57] Alexander Engels, Michael Reyer, Xiang Xu, Rudolf Mathar, Jietao Zhang, and Hongcheng Zhuang. “Autonomous Self-Optimization of Coverage and Capacity in LTE Cellular Networks”. In: *IEEE Transactions on Vehicular Technology* 62.5 (2013), pp. 1989–2004.

[58] Eren Balevi and Jeffrey Andrews. “Online Antenna Tuning in Heterogeneous Cellular Networks With Deep Reinforcement Learning”. In: *IEEE Transactions on Cognitive Communications and Networking* PP (Aug. 2019), pp. 1–1.

[59] Ryan M. Dreifuerst, Samuel Daulton, Yuchen Qian, Paul Varkey, Maximilian Balandat, Sanjay Kasturia, Anoop Tomar, Ali Yazdan, Vish Ponnampalam, and Robert W. Heath. “Optimizing Coverage and Capacity in Cellular Networks using Machine Learning”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 8138–8142.

[60] Filippo Vannella, Grigorios Iakovidis, Ezeddin Al Håkim, Erik Aumayr, and Saman Feghhi. “Remote Electrical Tilt Optimization via Safe Reinforcement Learning”. In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)* (2021), pp. 1–7.

[61] Yuanjie Lin, Hui Gao, Wenjun Xu, and Yueming Lu. “Dynamic Antenna Configuration for 3D Massive MIMO System via Deep Reinforcement Learning”. In: *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. 2020, pp. 1–6.

[62] *TS 36.214 - Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements*. Rel 17.0.0. 3GPP. Apr. 2022.

[63] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.

[64] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. “Deep reinforcement learning that matters”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[65] Hang Zou, Qiyang Zhao, Lina Bariah, Mehdi Bennis, and Merouane Debbah. “Wireless Multi-Agent Generative AI: From Connected Intelligence to Collective Intelligence”. In: *arXiv preprint arXiv:2307.02757* (2023).

[66] Helin Yang, Arokiaswami Alphones, Zehui Xiong, Dusit Niyato, Jun Zhao, and Kaishun Wu. “Artificial-Intelligence-Enabled Intelligent 6G Networks”. In: *IEEE Network* 34.6 (2020), pp. 272–280.

[67] Hoon Lee, Sang Hyun Lee, and Tony Q. S. Quek. "Artificial Intelligence Meets Autonomy in Wireless Networks: A Distributed Learning Approach". In: *IEEE Network* 36.6 (2022), pp. 100–107.

[68] Athanasios Karapantelakis, Pegah Alizadeh, Abdulrahman Alabassi, Kaushik Dey, and Alexandros Nikou. "Generative AI in mobile networks: a survey". In: *Annals of Telecommunications* (2023).

[69] Zhengwei Wang, Qi She, and Tomas E Ward. "Generative adversarial networks in computer vision: A survey and taxonomy". In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–38.

[70] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. "Improving language understanding by generative pre-training". In: (2018).

[71] Yinqiu Liu, Hongyang Du, Dusit Niyato, Jiawen Kang, Zehui Xiong, Dong In Kim, and Abbas Jamalipour. "Deep generative model and its applications in efficient wireless network management: A tutorial and case study". In: *arXiv preprint arXiv:2303.17114* (2023).

[72] Hojjat Navidan, Parisa Fard Moshiri, Mohammad Nabati, Reza Shahbazian, Seyed Ali Ghorashi, Vahid Shah-Mansouri, and David Windridge. "Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation". In: *Computer Networks* 194 (2021), p. 108149. URL: https://www.sciencedirect.com/science/article/pii/S1389128621002139.

[73] Ender Ayanoglu, Kemal Davaslioglu, and Yalin E Sagduyu. "Machine learning in nextg networks via generative adversarial networks". In: *IEEE Transactions on Cognitive Communications and Networking* 8.2 (2022), pp. 480–501.

[74] Cong Zou, Fang Yang, Jian Song, and Zhu Han. "Generative Adversarial Network for Wireless Communication: Principle, Application, and Trends". In: *IEEE Communications Magazine* (2023), pp. 1–7.

[75] Timothy J. O'Shea, Tamoghna Roy, and Nathan West. "Approximating the Void: Learning Stochastic Channel Models from Observation with Variational Generative Adversarial Networks". In: *2019 International Conference on Computing, Networking and Communications (ICNC)*. 2019, pp. 681–686.

[76] Yang Yang, Yang Li, Wuxiong Zhang, Fei Qin, Pengcheng Zhu, and Cheng-Xiang Wang. "Generative-Adversarial-Network-Based Wireless Channel Modeling: Challenges and Opportunities". In: *IEEE Communications Magazine* 57.3 (2019), pp. 22–27.

[77] Tribhuvanesh Orekondy, Arash Behboodi, and Joseph B Soriaga. “Mimo-gan: Generative mimo channel modeling”. In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 5322–5328.

[78] Ben Hughes, Shruti Bothe, Hasan Farooq, and Ali Imran. “Generative Adversarial Learning for Machine Learning empowered Self Organizing 5G Networks”. In: *2019 International Conference on Computing, Networking and Communications (ICNC)*. 2019, pp. 282–286.

[79] Paolo Di Francesco, Francesco Malandrino, and Luiz A. DaSilva. “Assembling and Using a Cellular Dataset for Mobile Network Analysis and Planning”. In: *IEEE Transactions on Big Data* 4.4 (2018), pp. 614–620.

[80] Chuanhao Sun, Kai Xu, Mahesh K. Marina, and Howard Benn. “GenDT: Mobile Network Drive Testing Made Efficient with Generative Modeling”. In: CoNEXT ’22. New York, NY, USA: Association for Computing Machinery, 2022.

[81] Jakob Thrane, Matteo Artuso, Darko Zibar, and Henrik L. Christiansen. “Drive Test Minimization Using Deep Learning with Bayesian Approximation”. In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. 2018, pp. 1–5.

[82] Songyang Zhang, Achintha Wijesinghe, and Zhi Ding. “RME-GAN: A Learning Framework for Radio Map Estimation Based on Conditional Generative Adversarial Network”. In: *IEEE Internet of Things Journal* 10.20 (2023), pp. 18016–18027.

[83] Danyang Zhang, Junhui Zhao, Lihua Yang, Yiwen Nie, and Xiangcheng Lin. “Generative Adversarial Network-based Channel Estimation in High-Speed Mobile Scenarios”. In: *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2021, pp. 1–5.

[84] Bassant Tolba, Maha Elsabrouty, Mubarak G. Abdu-Aguye, Haris Gacanin, and Hossam Mohamed Kasem. “Massive MIMO CSI Feedback Based on Generative Adversarial Network”. In: *IEEE Communications Letters* 24.12 (2020), pp. 2805–2808.

[85] Hao Han, Ximing Wang, Fanglin Gu, Wen Li, Yuan Cai, Yifan Xu, and Yuhua Xu. “Better Late Than Never: GAN-Enhanced Dynamic Anti-Jamming Spectrum Access With Incomplete Sensing Information”. In: *IEEE Wireless Communications Letters* 10.8 (2021), pp. 1800–1804.

[86] Suparna De, Maria Bermudez-Edo, Honghui Xu, and Zhipeng Cai. “Deep Generative Models in the Industrial Internet of Things: A Survey”. In: *IEEE Transactions on Industrial Informatics* 18.9 (2022), pp. 5728–5737.

[87] Mina Razghandi, Hao Zhou, Melike Erol-Kantarci, and Damla Turgut. "Smart Home Energy Management: VAE-GAN synthetic dataset generator and Q-learning". In: *arXiv preprint arXiv:2305.08885* (2023).

[88] Mina Razghandi, Hao Zhou, Melike Erol-Kantarci, and Damla Turgut. "Variational autoencoder generative adversarial network for Synthetic Data Generation in smart home". In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 4781–4786.

[89] Mustafizur R. Shahid, Gregory Blanc, Houda Jmila, Zonghua Zhang, and Hervé Debar. "Generative Deep Learning for Internet of Things Network Traffic Generation". In: *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. 2020, pp. 70–79.

[90] Aidin Ferdowsi and Walid Saad. "Generative Adversarial Networks for Distributed Intrusion Detection in the Internet of Things". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019, pp. 1–6.

[91] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. "The Synthetic Data Vault". In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2016, pp. 399–410.

[92] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. "PrivBayes: Private Data Release via Bayesian Networks". In: *ACM Trans. Database Syst.* 42.4 (Oct. 2017).

[93] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. "Modeling Tabular Data Using Conditional GAN". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[94] Stanislaw Wkeglarczyk. "Kernel density estimation and its application". In: *ITM Web of Conferences*. Vol. 23. EDP Sciences. 2018, p. 00037.

[95] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. "Uncertainty in Gradient Boosting via Ensembles". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=1Jv6b0Zq3qi.

[96] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. "A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions". In: *ACM computing surveys* 52.4 (Aug. 2019). URL: https://doi.org/10.1145/3343440.

[97] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. “Accurate uncertainties for deep learning using calibrated regression”. In: *International conference on machine learning*. PMLR. 2018, pp. 2796–2804.

[98] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. “A Survey of Deep Active Learning”. In: *ACM computing surveys* 54.9 (Oct. 2021).

[99] Daoud Burghal, Ashwin T Ravi, Varun Rao, Abdullah A Alghafis, and Andreas F Molisch. “A comprehensive survey of machine learning based localization with wireless signals”. In: *arXiv preprint arXiv:2012.11171* (2020).

[100] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H Vincent Poor. “Distributed learning in wireless networks: Recent progress and future challenges”. In: *IEEE Journal on Selected Areas in Communications* 39.12 (2021), pp. 3579–3605.

[101] Kam-Chuen Jim and C. Lee Giles. “How Communication Can Improve the Performance of Multi-Agent Systems”. In: *Proceedings of the Fifth International Conference on Autonomous Agents*. AGENTS ’01. Montreal, Quebec, Canada: Association for Computing Machinery, 2001, pp. 584–591. URL: https://doi.org/10.1145/375735.376455.

[102] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. “Relational inductive biases, deep learning, and graph networks”. In: *arXiv preprint arXiv:1806.01261* (2018).

[103] Michael L. Littman. “Markov Games as a Framework for Multi-Agent Reinforcement Learning”. In: *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*. ICML’94. New Brunswick, NJ, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 157–163.

[104] Sainbayar Sukhbaatar, Rob Fergus, et al. “Learning multiagent communication with backpropagation”. In: *Advances in neural information processing systems* 29 (2016).

[105] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. “Cooperative multi-agent control using deep reinforcement learning”. In: *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*. Springer. 2017, pp. 66–83.

[106] Mark Eisen and Alejandro Ribeiro. “Optimal wireless resource allocation with random edge graph neural networks”. In: *IEEE transactions on signal processing* 68 (2020), pp. 2977–2991.

[107] Zhan Gao, Yulin Shao, Deniz Gunduz, and Amanda Prorok. "Decentralized channel management in WLANs with graph neural networks". In: *arXiv preprint arXiv:2210.16949* (2022).

[108] Kota Nakashima, Shotaro Kamiya, Kazuki Ohtsu, Koji Yamamoto, Takayuki Nishio, and Masahiro Morikura. "Deep reinforcement learning-based channel allocation for wireless lans with graph convolutional networks". In: *IEEE Access* 8 (2020), pp. 31823–31834.

[109] Navid NaderiAlizadeh, Mark Eisen, and Alejandro Ribeiro. "Adaptive Wireless Power Allocation with Graph Neural Networks". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 5213–5217.

[110] Xiaochen Zhang, Haitao Zhao, Jun Xiong, Xiaoran Liu, Li Zhou, and Jibo Wei. "Scalable power control/beamforming in heterogeneous wireless networks with graph neural networks". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2021, pp. 01–06.

[111] Bohan Li, Lie-Liang Yang, Robert G Maunder, Songlin Sun, and Pei Xiao. "Heterogeneous graph neural network for power allocation in multicarrier-division duplex cell-free massive MIMO systems". In: *IEEE Transactions on Wireless Communications* (2023).

[112] Jiechuan Jiang and Zongqing Lu. "Learning attentional communication for multi-agent cooperation". In: *Advances in neural information processing systems* 31 (2018).

[113] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. "Tarmac: Targeted multi-agent communication". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1538–1546.

[114] Michael I. Jordan. "Graphical Models". In: *Statistical Science* 19.1 (2004), pp. 140–155. URL: https://doi.org/10.1214/088342304000000026.

[115] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[116] Judea Pearl. "Causal Inference". In: *Proceedings of Workshop on Causality: Objectives and Assessment at NIPS 2008*. Ed. by Isabelle Guyon, Dominik Janzing, and Bernhard Schölkopf. Vol. 6. Proceedings of Machine Learning Research. Whistler, Canada: PMLR, Dec. 2010, pp. 39–58.

[117] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[118] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. "Weisfeiler and leman go neural: Higher-order graph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

[119] *TS 38.901 - Study on channel model for frequencies from 0.5 to 100 GHz*. Rel 15.0.0. 3GPP. July 2018.

[120] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. "Next Generation 5G Wireless Networks: A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 18.3 (2016), pp. 1617–1655.

[121] Lalit Chettri and Rabindranath Bera. "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems". In: *IEEE Internet of Things Journal* 7.1 (2020), pp. 16–32.

[122] Kinza Shafique, Bilal A. Khawaja, Farah Sabir, Sameer Qazi, and Muhammad Mustaqim. "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios". In: *IEEE Access* 8 (2020), pp. 23022–23040.

[123] Miaowen Wen, Qiang Li, Kyeong Jin Kim, David López-Pérez, Octavia A. Dobre, H. Vincent Poor, Petar Popovski, and Theodoros A. Tsiftsis. "Private 5G Networks: Concepts, Architectures, and Research Landscape". In: *IEEE Journal of Selected Topics in Signal Processing* 16.1 (2022), pp. 7–25.

[124] 5G-ACIA. *5G for Connected Industries and Automation, Second Edition*. Feb. 2019.

[125] 5G-ACIA. *5G for Industrial Internet of Things (IIoT): Capabilities, Features, and Potential*. Nov. 2021.

[126] Aamir Mahmood, Luca Beltramelli, Sarder Fakhrul Abedin, Shah Zeb, Nishat I. Mowla, Syed Ali Hassan, Emiliano Sisinni, and Mikael Gidlund. "Industrial IoT in 5G-and-Beyond Networks: Vision, Architecture, and Design Trends". In: *IEEE Transactions on Industrial Informatics* 18.6 (2022), pp. 4122–4137.

[127] Senthil Kumar Jagatheesaperumal, Mohamed Rahouti, Kashif Ahmad, Ala Al-Fuqaha, and Mohsen Guizani. "The Duo of Artificial Intelligence and Big Data for Industry 4.0: Applications, Techniques, Challenges, and Future Research Directions". In: *IEEE Internet of Things Journal* 9.15 (2022), pp. 12861–12885.

[128] Martin Müller, Fjolla Ademaj, Thomas Dittrich, Agnes Fastenbauer, Blanca Elbal, Armand Nabavi, Lukas Nagel, Stefan Schwarz, and Markus Rupp. "Flexible multi-node simulation of cellular mobile communications: the Vienna 5G System Level Simulator". In: *EURASIP Journal on Wireless Communications and Networking* (2018).

[129] Lianfen Huang, Tao Chen, Zhibin Gao, Manman Luo, and Zhang Liu. “System Level simulation for 5G Ultra-Reliable Low-Latency Communication”. In: *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. 2021, pp. 1–5.

[130] George F. Riley and Thomas R. Henderson. “The ns-3 Network Simulator”. In: *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010, pp. 15–34. URL: https://doi.org/10.1007/978-3-642-12331-3_2.

[131] Marco Mezzavilla, Menglei Zhang, Michele Polese, Russell Ford, Sourjya Dutta, Sundeep Rangan, and Michele Zorzi. “End-to-End Simulation of 5G mmWave Networks”. In: *IEEE Communications Surveys & Tutorials* (May 2017).

[132] Giampaolo Cuozzo, Sara Cavallero, et al. “Enabling URLLC in 5G NR IIoT Networks: A Full-Stack End-to-End Analysis”. In: *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. June 2022, pp. 333–338.

[133] Yong Zhao, Mingshuo Wei, Chunlei Hu, and Weiliang Xie. “Latency Analysis and Field Trial for 5G NR”. In: *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 2022, pp. 1–5.

[134] Justus Rischke, Peter Sossalla, Sebastian Itting, Frank H. P. Fitzek, and Martin Reisslein. “5G Campus Networks: A First Measurement Study”. In: *IEEE Access* (2021).

[135] Junaid Ansari, Christian Andersson, Peter de Bruin, János Farkas, Leefke Grosjean, Joachim Sachs, Johan Torsner, Balázs Varga, Davit Harutyunyan, Niels König, and Robert H. Schmitt. “Performance of 5G Trials for Industrial Automation”. In: *Electronics* (2022). URL: https://www.mdpi.com/2079-9292/11/3/412.

[136] Winnie Nakimuli, Jaime Garcia-Reinoso, J. Enrique Sierra-Garcia, Pablo Serrano, and Isaac Quintana Fernández. “Deployment and Evaluation of an Industry 4.0 Use Case over 5G”. In: *IEEE Communications Magazine* (2021), pp. 14–20.

[137] Stanislav Vakaruk, J. Enrique Sierra-García, Alberto Mozo, and Antonio Pastor. “Forecasting Automated Guided Vehicle Malfunctioning with Deep Learning in a 5G-Based Industry 4.0 Scenario”. In: *IEEE Communications Magazine* 59.11 (2021), pp. 102–108.

[138] Lei Ren, Yaqiang Sun, Jin Cui, and Lin Zhang. “Bearing remaining useful life prediction based on deep autoencoder and deep neural networks”. In: *Journal of Manufacturing Systems* (2018). Special Issue on Smart Manufacturing, pp. 71–77. URL: https://www.sciencedirect.com/science/article/pii/S0278612518300475.

[139] Rui Zhao, Jinjiang Wang, Ruqiang Yan, and Kezhi Mao. "Machine health monitoring with LSTM networks". In: *2016 10th International Conference on Sensing Technology (ICST)*. IEEE. 2016, pp. 1–6.

[140] Felix O Heimes. "Recurrent neural networks for remaining useful life estimation". In: *2008 International Conference on Prognostics and Health Management*. IEEE. 2008, pp. 1–6.

[141] Xiang Li, Qian Ding, and Jian-Qiao Sun. "Remaining useful life estimation in prognostics using deep convolution neural networks". In: *Reliability Engineering & System Safety* (2018), pp. 1–11. URL: https://www.sciencedirect.com/science/article/pii/S0951832017307779.

[142] Kwangsuk Lee, Jae-Kyeong Kim, Jaehyong Kim, Kyeon Hur, and Hagbae Kim. "CNN and GRU combination scheme for bearing anomaly detection in rotating machinery health monitoring". In: *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*. IEEE. 2018, pp. 102–105.

[143] Rui Zhao, Ruqiang Yan, Jinjiang Wang, and Kezhi Mao. "Learning to monitor machine health with convolutional bi-directional LSTM networks". In: *Sensors* (2017), p. 273.

[144] Beata Mrugalska. "Remaining Useful Life as Prognostic Approach: a Review". In: *Human Systems Engineering and Design*. 2019, pp. 689–695.

[145] Hassan Ismail Fawaz, Germain Forestier, et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* (Mar. 2019), pp. 917–963. URL: https://doi.org/10.1007%2Fs10618-019-00619-1.

[146] Andrea Borghesi, Andrea Bartolini, et al. "Anomaly Detection Using Autoencoders in High Performance Computing Systems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* (July 2019), pp. 9428–9433. URL: https://doi.org/10.1609%2Faaai.v33i01.33019428.

[147] Yuting Wu, Mei Yuan, et al. "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks". In: *Neurocomputing* (2018), pp. 167–179. URL: https://www.sciencedirect.com/science/article/pii/S0925231217309505.

[148] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[149] 3GPP. *Study on channel model for frequencies from 0.5 to 100 GHz*. Technical Report (TR). Version 16.1.0 Release 16. 3rd Generation Partnership Project (3GPP), Nov. 2020. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173.

[150] Sara Cavallero, Nicol Sarcone Grande, Francesco Pase, Marco Giordani, Joseph Eichinger, Roberto Verdone, and Michele Zorzi. “A new scheduler for URLLC in 5G NR IIoT networks with spatio-temporal traffic correlations”. In: *ICC 2023-IEEE International Conference on Communications*. 2023, pp. 1010–1015.

[151] Guo Haixiang, Li Yijing, et al. “Learning from class-imbalanced data: review of methods and applications”. In: *Expert Systems with Applications* (2017), pp. 220–239. URL: https://www.sciencedirect.com/science/article/pii/S0957417416307175.

[152] Jason Brownlee. *Introduction to Time Series Forecasting with Python*. Machine Learning Mastery, 2018.

[153] Radu Manuca and Robert Savit. “Stationarity and nonstationarity in time series analysis”. In: *Physica D: Nonlinear Phenomena* (1996), pp. 134–161. URL: https://www.sciencedirect.com/science/article/pii/S016727899600139X.

[154] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. “Language models are few-shot learners”. In: *Advances in Neural Information Processing Systems* (2020), pp. 1877–1901.

[155] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. “Are transformers effective for time series forecasting?” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2023, pp. 11121–11128.