Shared-resource Generative Adversarial Network (GAN) Training for 5G Ultra Reliable Low Latency Communication (URLLC) Deep Reinforcement Learning Augmentation

by

Kaveh MEHDIPOURCHARI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M.A.Sc.

MONTREAL, AUGUST 27, 2024

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Kim Khoa Nguyen, Thesis supervisor
Department of Electrical Engineering, Ecole de technologie superieure

Mr. Lokman Sboui, Chair, Board of Examiners
Department of System Engineering, Ecole de technologie superieure

Mrs. Bassant Selim, External Examiner
Department of System Engineering, Ecole de technologie superieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON AUGUST 12, 2024

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I would like to thank my supervisor, who has supported me in every aspect over the past four years to help me achieve this valuable degree. I am deeply grateful to my family for their unwavering support and encouragement. Their belief in my abilities, numerous sacrifices, and constant encouragement have been the cornerstone of my journey to completing this challenging master's thesis.

Partage des Réseaux Antagonistes Génératifs (GAN) pour l'Augmentation de l'Apprentissage par Renforcement Profond dans les Communications 5G Ultra Fiables et à Faible Latence (URLLC)

Kaveh MEHDIPOURCHARI

RÉSUMÉ

Récemment, l'apprentissage par renforcement profond (DRL) a démontré sa capacité en résolvant des problèmes hautement complexes en temps réel. Cependant, le DRL rencontre des difficultés en gérant les événements rares dans les communications sans fil, principalement parce que les agents DRL manquent d'expérience avec de telles situations pendant leur entraînement. Ce défi est particulièrement crucial pour les communications ultra-fiables et à faible latence (URLLC) de la 5G, où les événements rares peuvent diminuer significativement la fiabilité de communication.

Pour résoudre ce problème, je me suis concentré sur l'utilisation de techniques d'augmentation de données, surtout les réseaux antagonistes génératifs (GANs), pour permettre aux agents DRL d'apprendre à gérer les événements rares en les exposant à une gamme diversifiée de scénarios. Cependant, l'entraînement des GANs dans les réseaux 5G avec des ressources limitées, comme les réseaux ultra-denses (UDNs), est un défi difficile puisque ces réseaux n'ont pas suffisamment de puissance de calcul pour traiter des données de haute dimension. Malgré son importance, il n'y avait pas encore beaucoup de travaux antécédents portant sur l'entraînement efficace des GANs dans des environnements à ressources limitées.

Dans ce mémoire, je conçois une nouvelle architecture et présente un problème d'optimisation pour entraîner des GANs dans des réseaux 5G à ressources limitées. L'architecture proposée partage efficacement des ressources informatiques et de bande passante limitées entre les environnements de cloud et de périphérie pour entraîner plusieurs GANs simultanément. Ensuite, les données augmentées provenant des GANs sont combinées avec des données réelles pour entraîner des modèles DRL. Mon modèle mathématique appelé OGAN, conçu pour optimiser l'allocation des ressources de calcul et de communication pour entraîner des GANs, afin d'améliorer la fiabilité du DRL, un aspect critique des URLLC.

Puisque OGAN est un problème mixte en nombres entiers non convexe difficile, je l'approximons en utilisant l'approche de programmation par différence de fonctions convexes et le résous avec un algorithme convexe-concave. Enfin, je réalise des expérimentations extensibles pour évaluer sa performance par rapport à deux méthodes de référence.

Mots-clés: GAN, URLLC, Apprentissage par renforcement profond, Allocation des ressources

Shared-resource Generative Adversarial Network (GAN) Training for 5G Ultra Reliable Low Latency Communication (URLLC) Deep Reinforcement Learning Augmentation

Kaveh MEHDIPOURCHARI

ABSTRACT

Recently, Deep Reinforcement Learning (DRL) has demonstrated exceptional abilities in finding high-quality real-time solutions. Nonetheless, DRL encounters difficulties in managing rare events in wireless communications, primarily because DRL agents lack experience with such situations during their training. This challenge is particularly crucial for 5G ultra-reliable low-latency communications (URLLC), where rare events can significantly reduce communication reliability.

To address this issue, recent efforts have focused on utilizing data augmentation techniques, such as Generative Adversarial Networks (GANs), to enable DRL agents to learn to handle rare events by exposing them to a diverse range of scenarios. However, training GANs in resource-constrained 5G networks, like Ultra Dense Networks (UDNs), remains a challenging task due to the substantial computational power required, especially when processing high-dimensional data. Despite its importance, there has been limited attention on effectively training GANs within resource-limited environments.

In this thesis, I introduce a novel architecture and an optimization problem for training GANs in resource-constrained 5G networks. My proposed architecture efficiently shares limited computing and bandwidth resources between the cloud and edge environments for training multiple GANs. Then, the augmented datasets from the trained GANs are combined with real datasets to be used for training DRL models. I present a mathematical model called OGAN, designed to optimize the allocation of computation and communication resources for GAN training, aiming to enhance DRL reliability, a critical aspect of URLLC. Since OGAN is a challenging mixed-integer non-convex problem, I approximate it using the Difference of Convex Functions programming approach and solve it with a Convex-Concave Algorithm. Finally, I carry out extensive simulations to evaluate OGAN's performance in comparison with two baseline methods.

Keywords: URLLC, GAN, Deep reinforcement learning, Resource allocation

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

Page

LIST OF ABBREVIATIONS

URLLC	Ultra-Reliability and Low-Latency Communication
5G	Fifth-Generation
IoT	Internet of Things
QoS	Quality-of-Service
DRL	Deep Reinforcement Learning
GAN	Generative Adversarial Network
UDN	Ultra-Dense Network
DNN	Deep Neural Network
SBS	Small Base Station
MEC	Mobile Edge Computing
OFDMA	Orthogonal Frequency-Division Multiple Access
RB	Resource Block
SO	Sub-Objective
MDP	Markov Decision Processes
eMBB	Enhanced Mobile BroadBand
Ge	Generator
Di	Discriminator
DCGAN	Deep Convolutional Generative Adversarial Network
UAV	Unmanned Aerial Vehicle

mmWave	Millimeter Wave
MIMO	Multiple Input Multiple Output
BS	Base Station
ML	Machine Learning
RL	Reinforcement Learning
GPU	Graphics Processing Unit
CPU	Central Processing Unit

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

bps	Bits per second
s	Second

INTRODUCTION

Context and motivations

One of the main features of 5G networks is their support for Ultra-Reliable and Low Latency Communications (URLLC), which is essential for achieving exceptional reliability and low latency in end-to-end communication. This new feature of the 5G technology paves the way for numerous potential and exciting applications, including autonomous vehicles, industrial automation, massive machine-type communications, the Internet of Things (IoT), and virtual/augmented reality. To support URLLC services, several innovative specifications have been introduced in 5G New Radio, in particular flexible numerology which allows for the use of various sub-carrier spacings, resulting in different lengths of Orthogonal Frequency-Division Multiple Access (OFDMA) symbols with slot durations starting from 125 μs(Ali, Zikria, Bashir, Garg & Kim (2021)).

The complexity of 5G URLLC and its strict time constraints make wireless resource allocation a challenging task particularly to find optimal solutions using traditional mathematical methods (Meng, Chen, Wu & Cheng (2020)). Model-based methods have proven very useful for performance analysis and network optimization in previous generations of mobile networks. To achieve tractable solutions, model-based methods are not sufficient for dealing with the challenges of next-generation networks such as URLLC services, since they require some ideal assumptions and simplifications. Therefore, the obtained solutions cannot satisfy the requirements of real-world networks with respect to quality of service (QoS). In addition, many of the optimization problems proposed for resource allocation, scheduler design, and network management are non-convex in nature. Optimizing wireless network policies based on dynamic parameters requires that the system executes searching algorithms every few milliseconds. This results in high computing overheads and long processing times, which is not ideal for URLLC (She *et al.* (2020)).

An expected benefit of 5G networks is their ability in supporting a wide variety of applications with diverse requirements, such as higher user data rates, improved energy efficiency, reduced latency, and enhanced reliability (Navarro-Ortiz *et al.* (2020)). However, the 5G network environment is dynamically uncertain, characterized by fast-changing wireless channels and constantly shifting topologies. Therefore, optimizing network resource and service management is a challenging task that requires new approaches (Ye *et al.* (2018)). Deep Reinforcement Learning (DRL) stands out as a powerful solution to meet the evolving demands of 5G networks, offering novel strategies for adaptive resource allocation and dynamic network optimization. It is able to quickly learn the optimal policies with minimal computing overheads.

DRL has proven to be an effective methodology for addressing real-time dynamic decision-making problems in time-varying and unpredictable wireless networks (Xiong *et al.* (2019)). It incorporates farsighted system evolution rather than myopically optimizing current benefits, a quality essential for time-variant 5G networks. Even without current information, DRL can update decision policies to optimize system performance through feedback based on previous decisions. Therefore, DRL-based approaches are a prominent option for resource and service management in large-scale mobile networks, particularly when there is a need for real-time decisions such as URLLC, where the transmissions delay is a few millisecond (Wang, Wang, Hao, Wu & Poor (2020)).

The remarkable capabilities of DRL in finding high-quality solutions in real-time make it a viable candidate for radio resource allocation in 5G URLLC networks. Adaptive learning, a key feature of DRL, allows it to optimize network performance and enhance resource allocation in dynamic wireless environments. By leveraging DRL's ability to learn from experience and make intelligent decisions, URLLC networks can achieve improved efficiency, reduced latency, and enhanced overall reliability. Deploying DRL in this context demonstrates its versatility and positions it as a cutting-edge solution for the intricate service demands of next-generation

wireless networks (Xiong *et al.* (2019)). However, despite its superior performance in research, real-world implementation encounters challenges, such as communication unreliability due to rare events in its state-space (Dulac-Arnold *et al.* (2021)).

Training Reinforcement learning for URLLC

A key challenge in implementing DRL in critical systems (e.g. URLLC resource management) is its unreliability when it is directly trained in the environment. Unlike supervised learning, which requires the burden of labeling, DRL can be trained in an environment with minimum human supervision. During the training phase, the DRL agent interacts with its environment to discover the best weights for its DNN (Berkenkamp, Turchetta, Schoellig & Krause (2017)). This process, called exploration, involves the agent experimenting with different actions to gather information and identify actions that maximize long-term rewards. However, excessive exploration can lead to suboptimal performance as the agent may spend too much time trying less promising actions instead of using its current knowledge to optimize immediate rewards.

In the context of URLLC resource allocation, this exploration can lead to the DRL agent initially making suboptimal allocation decisions. Suboptimal decisions can be manifested in two ways: overestimation of resources and underestimation of resources. In the former case, the agent allocates more resources than necessary, resulting in resource waste and inefficiency. Alternatively, in the latter situation, the agent may not allocate enough resources, leading to reduced user satisfaction and QoS degradation (Kasgari, Saad, Mozaffari & Poor (2020)). In critical systems such as robotics and autonomous driving, packet loss and transmission delays (QoS degradation) pose significant safety hazards and operational inefficiencies.

Pre-training DRL in a virtual environment

Pretraining in a simulated environment is a common strategy to mitigate the risks of suboptimal actions and enhance DRL agent performance. During exploration, trial-and-error interactions

with the environment are crucial for learning, but they can be time-consuming and costly, especially in resource-constrained networks. Additionally, directly training a DRL agent in the real world, particularly in safety-critical applications, raises concerns about potentially encountering unforeseen and potentially dangerous behaviors during training (Ho, Nguyen, Nguyen & Cheriet (2021)). Therefore, pretraining offers a valuable solution by allowing the agent to learn fundamental behaviors in a controlled environment, followed by fine-tuning or transfer learning to the real world for adaptation to specific nuances.

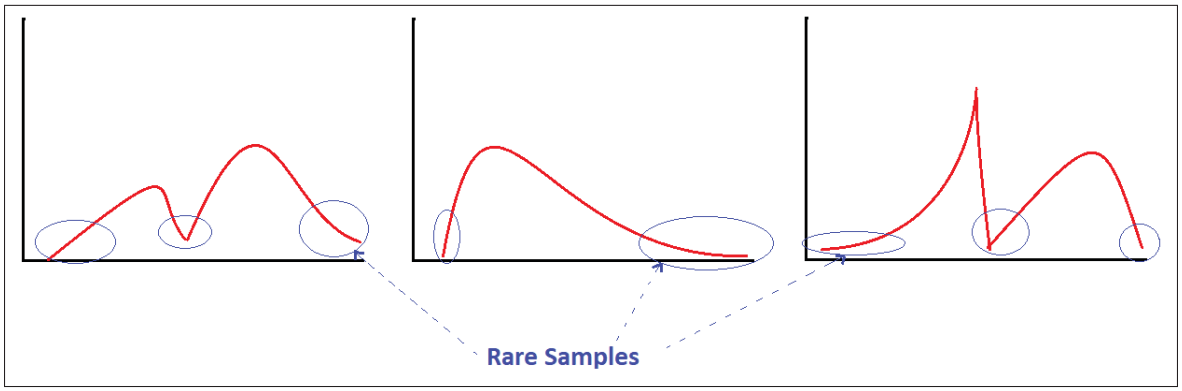

Figure 0.1 Examples of rare events across non-uniform data distributions. The red curves represent the probability density functions, and the blue circles highlight the rare samples. These rare samples, found in the tails of the distributions, are critical for the performance of a DRL agent, presenting challenges in safety-critical systems.

Generally, DRL encounters significant challenges when addressing rare events and uneven state-space distribution. Some examples of these challenges are shown in Figure 0.1. Rare events, often sparsely distributed across the state-space, make it difficult for DRL agents to experience them sufficiently during training. Examples in a wireless network include deep fades, interference, system errors, and traffic anomalies. The uneven distribution of datasets from real-world environments leads to DRL agents predominantly learning from more common events, which have higher probabilities of occurrence (Chan, Lampinen, Richemond & Hill (2022)). This focus on majority classes results in suboptimal performance when agents encounter rare

events in real wireless environments, ultimately causing Quality of Service (QoS) degradation. These rare events are crucial for the model to perform well in real-world scenarios, but their infrequent occurrence means the model doesn't get enough exposure to them during training, leading to poor performance when such events do occur.

In URLLC, the importance of tackling the challenge of rare events for DRL is greater, since they can lead to communication unreliability, manifesting as issues such as packet loss and transmission delays. Since DRL agent used for resource allocation does not have enough experience with rare events, they may underestimate the required resources (e.g. bandwidth, transmission power). As a result, transmission delay and packet loss decreases the reliability of the network as well as increasing the re-transmission. The stringent requirements of URLLC in terms of reliability and low-latency performance, emphasize the importance of increasing the capability of DRL models responding to rare events.

A common way to boost the DRL's generalization ability, enabling them to make better decisions when faced with rare events is using broader and more diverse training set. This way, by exposing them to different environment scenarios, including rare events, DRL agents have a more robust understanding of the underlying dynamics of the environment. Consequently, they gain the ability to navigate the complexities of real-world applications, ensuring more reliable and adaptable performance in dynamic and evolving environments. However, in cases such as safety critical systems, gathering a huge dataset of rare events is either expensive or impossible. As wireless data is influenced by the environment, training data generated in a laboratory setting may not accurately represent real-world conditions (Davaslioglu & Sagduyu (2018)). Thus, it is crucial to ensure that the training data reflects the actual environment where the wireless data will be utilized. Using generative models, such as GANs, to augment datasets with rare real-world events is a practical approach to enhance dataset diversity while maintaining environmental characteristics (Ayanoglu, Davaslioglu & Sagduyu (2022)).

To address the challenge of handling rare events in DRL, recent advancements utilize GANs to increase the diversity of the training dataset. GANs generate synthetic examples of rare events based on real occurrences and integrate them into the DRL training dataset. Compared to other generative models, GANs require less computational power and human intervention while producing higher-quality and more diverse datasets. By using GANs, I can enhance the representation of rare events, which include significant but uncommon scenarios crucial for performance and reliability in real-world applications. This augmentation helps DRL models improve their ability to handle rare events, overcoming the limitations posed by insufficient exposure during training (Kasgari *et al.* (2020)).

Challenges

The small cell concept plays a crucial role in achieving the high demands of 5G cellular networks by increasing network capacity and improve data rates (throughput). In such networks, known as ultra dense networks (UDN), a large number of small cells need to be deployed densely, estimating 40-50 base stations per square kilometer (BSs/km^2). Consequently, 5G networks are moving towards a paradigm shift, transitioning from the traditional macrocell-based architecture to an UDN where SBSs play a dominant role in providing ubiquitous and high-performance coverage (Wang *et al.* (2014)).

While UDN offers significant advantages in 5G networks, they also introduce new challenges related to computation and backhaul bandwidth limitations. To be cost efficient, SBSs have limited network resources and processing power compared to macrocells. This can hinder their ability to handle the complex processing tasks such as training large DNN models such as GAN. Backhaul bandwidth, which refers to the data transfer capacity between SBSs and the core network, also presents a significant constraint. Densely deployed SBS networks generate a large volume of data traffic, and the backhaul infrastructure needs to be sufficiently robust to handle

this increased load. Insufficient backhaul capacity can lead to congestion, delays, and ultimately, a degraded user experience (Ge, Tu, Mao, Wang & Han (2016)).

Prior research on GANs in 5G networks, as reviewed by (Navidan *et al.* (2021)), has primarily focused on proposing novel DNN architectures and loss models for GANs to address challenges related to tasks like data augmentation and learning complex data distributions. Little attention has been paid to training GANs in resource-constrained environments, such as 5G UDNs. High node density in UDNs presents the following challenges for applying GANs:

- SBSs have very limited computational power, insufficient for training resource-intensive DNNs like GANs.
- Utilizing the cloud or mobile edge computing (MEC) for training DNN models presents challenges due to limited backhaul connection capacity. As SBSs typically share limited bandwidth wireless backhaul, it might not be feasible to transmit all training datasets simultaneously.
- The limited processing power of edge and cloud resources hinders the simultaneous and timely training of GANs for all SBSs.

To allocate resources as optimally as possible for a UDN, an effective mechanism is required for selecting the optimal subsets of SBSs for the timely training of GANs and DRL models. Traditional resource allocation strategies in machine learning—such as those focused on deadlines, importance, or time fairness—often fail to account for the diverse resource demands of GANs, particularly in terms of data and computation (Sun & et al. (2022)). This oversight can result in suboptimal performance and inefficiencies. Specifically, GANs require substantial computational power and extensive datasets to generate realistic synthetic examples, which are crucial for improving DRL training. Since conventional methods cannot address these heterogeneous resource requirements efficiently, they inadvertently limit the effectiveness of the training process, leading to low system performance.

Factors like channel characteristics and user behavior change frequently in 5G networks, affecting the performance of DNNs. When the environment changes, a trained model may become outdated and perform poorly which requires being retrained with new data from the includes environment. This includes retraining of both DRL models to adapt them to the environment change as well as GAN models for the new environment. This process is computationally expensive and time-consuming, hindering the network's ability to adapt and deliver optimal performance. In addition the training of GAN and DRLs should be done as quickly as possible to meet delay requirements of 5G. A long training time for DRL or GAN makes these models less effective because their training data can quickly become outdated.

Research questions

GANs are a promising solution to enhance the generalization and reliability of 5G URLLC by augmenting the training datasets for DRL. However, training GANs within UDNs presents significant challenges due to their high computational power requirements, especially in complex, high-dimensional environments. The main challenges reside in resource constraints, which does not allow multiple SBSs in a UDN to train their GANs concurrently. Therefore, an effective mechanism is required to allocate resources efficiently and to select the subset of SBSs that would benefit most from GAN training, thus maximizing overall system reliability. This thesis aims to address the following research question:

QA- How to allocate resources optimally for both GAN and DRL training in a resource constrained UDN environment to maximize DRL reliability?

Objectives of the thesis

Our primary goal is to define a model to optimize resource allocation for training both GAN and DRL for resource-constrained UDNs to minimize unreliability arising from DRL. This objective can be divided into the following three sub-objectives (SOs):

- **SO1** Design an architecture to share network and computing resources of a UDN, including both cloud and edge parts, for training GANs, and then apply the augmented datasets for training DRLs.
- **SO2** Formulate an optimization model to allocate resources for training GANs, aiming to enhance DRL reliability. Such an optimization model should take into account the requirements of DRLs and GANs, particularly the GAN quality.
- **SO3** Propose an efficient solution to solve my optimization model so that meeting the constraints of 5G in terms of time.

Thesis organization

This thesis is organized into four main chapters, with a conclusion. The chapters are arranged as follows:

- **Chapter 1 (Introduction):** This chapter sets the stage by introducing the context and motivation for the study, focusing on the application of DRL in 5G URLLC networks. It outlines the challenges faced by DRL in this context and application of GAN for solving these challenges. Finally, I present the challenges of training and using GAN followed by the problem statement, research questions, and objectives.
- **Chapter 2 (Technical Background and Related Work):** In this chapter, I provide a detailed overview of the technical background relevant to my research, including reinforcement learning, DRL models, and GANs. I also review previous studies on the application of GANs in 5G networks and discuss various frameworks and architectures for training machine learning models in 5G edge networks.
- **Chapter 3 (Methodology):** Here, I present my proposed methodology. The first part is dedicated to system modeling, describing the system model of a 5G Ultra-Dense Network (UDN) and the proposed architecture for training GAN/DRL. The second part introduces the optimization model I propose, along with the algorithm developed to find its solution.

- **Chapter 4 (Experimental Setup and Results):** In this chapter, I detail the experimental setup and simulation scenarios. Then, I present the simulation results and provide a thorough discussion of the findings.
- **Conclusion:** The concluding section summarizes the key findings of the thesis, discusses their implications, and suggests directions for future research.

CHAPTER 1

LITERATURE REVIEW

In this section I provide the technical background and related works. I consider 5G URLLC, deep reinforcement learning and its application in 5G, generative adversarial network (GAN) and its application in 5G networks. I then review the literature according to the research question.

1.1 Background

1.1.1 Deep Reinforcement Learning

In this subsection I explain the basic theory of reinforcement learning (RL) and its training process following with deep RL (DRL). Then I present the application of DRL in 5G networks following by its challenges for URLLC.

RL is a machine learning model wherein the decision maker (called an agent) learns a policy to maximize a long-term reward by interacting with the environment. The primary distinguishing characteristic of RL is the trial-and-error learning process. In the learning process, the agent explores various actions, receives feedback from the environment in the form of rewards or penalties. Finally the agent adjusts its decision-making strategy accordingly to maximize the cumulative reward over the course of multiple interactions with the environment. This learning process is guided by the agent's policy, a set of guidelines or tactics that dictate the actions the agent ought to take in various situations of the environment. Unlike supervised learning, RL doesn't require a labelled dataset or explicit programming of decision rules, making it ideal for situations where the best course of action is not known beforehand. RL has demonstrated success in a wide range of applications, including control, autonomous systems, and optimization problems. Due to its adaptability and ability to handle complex, dynamic environments, it is a powerful tool for research and practical applications in various fields (Kiumarsi, Vamvoudakis, Modares & Lewis (2017)).

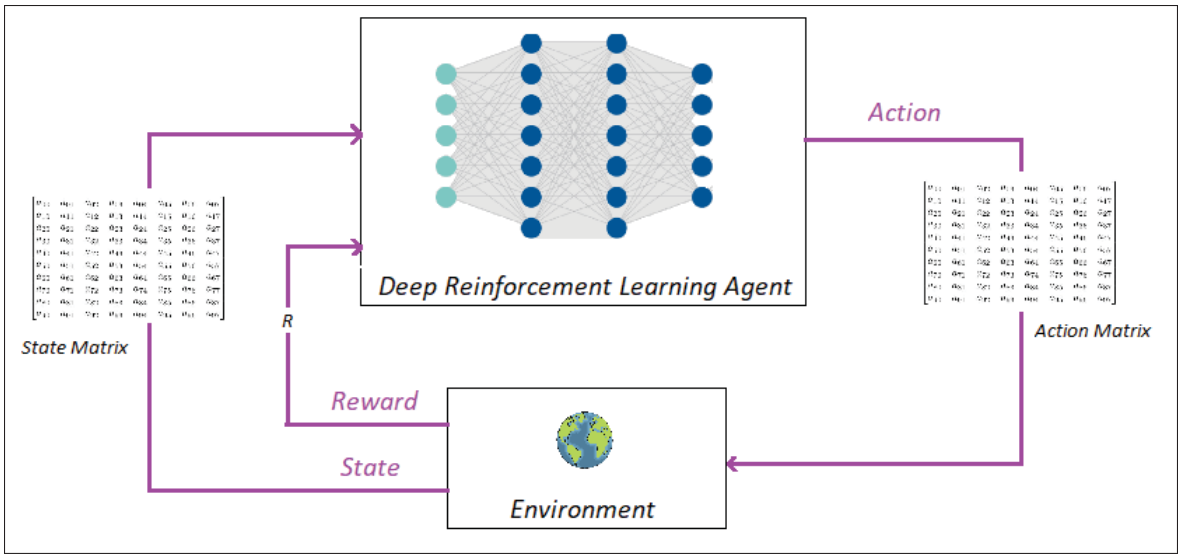

Figure 1.1 A DRL framework where the agent interacts with the environment. The state matrix is fed into the DRL agent, which processes the information and generates an action matrix. The environment receives the action, resulting in a new state and a reward (R) that are fed back to the agent, forming a continuous learning loop

Standard reinforcement learning tasks involve sequential decision-making, where an agent interacts with an environment one step at a time. At each time step t, as shown in Figure 1.1, the agent takes action a_t, which causes the environment to transition from state s_t to a new state s_{t+1} with a probability of $p(s_{t+1}|s_t, a_t)$. This setting is characterized as a Markov Decision Process (MDP), formally defined by the tuple $\mathbf{D} = (\mathcal{S}, \mathcal{A}, P, R)$, which includes the following components:

1. States ($\mathcal{S}$): A set of environment and agent states that represents all possible situations or configurations in the environment.

2. Actions ($\mathcal{A}$): A set of actions that an agent can take in any given state.

3. Transition Probabilities (P): A set of transition probabilities $P(s, a, s')$ that describe the probability of transitioning from state s to state s' given action a. This function captures the stochastic nature of the environment.

4. Reward Function (R): An immediate or instantaneous reward function $R(s, a, s')$ that quantifies the immediate payoff or cost associated with taking action a in state s and transitioning to state s'.

Limitations of RL hinder its effectiveness in tackling large and complex problems causes the emergence of deep RL (DRL). In conventional reinforcement learning, agents are required to store optimal values of the State-Value Function, Action-Value Function (Q-function), and Policy, among others.Especially in scenarios with vast state and action spaces, there is a significant memory requirement for storing these values, along with slow convergence speeds due to the need to visit all states and try all actions to find the optimal policy, which can decrease performance. DRL emerges as a promising solution, combining the fundamentals of RL with the powerful capabilities of deep learning models. DRL empowers models to learn complex representations of intricate environments, enabling them to handle large state and action spaces more effectively. This approach has demonstrated remarkable success in various applications, including natural language processing and robotics and computer games (Mnih *et al.* (2015)).

1.1.2 GAN

The purpose of this subsection is to present GAN and describe the learning and training mechanisms of it. Later I present some applications of GANs in 5G for data augmentation.

GANs consist of two neural networks as shown in Figure 1.2: a generator (Ge) and a discriminator (Di), engaged in a minimax game. The generator aims to produce synthetic data samples from random noise (z) that are indistinguishable from authentic data samples (x). The discriminator endeavors to differentiate between real and generated samples. During training, Ge learns to generate new data samples that closely mimic the distribution of the real samples in its training dataset. Di acts as a binary classifier, attempting to distinguish between real samples and the fake samples produced by Ge, providing adversarial feedback. This feedback helps Ge improve its ability to generate realistic samples (Gui, Sun, Wen, Tao & Ye (2021)). After training, only Ge is used to produce new synthetic data samples, while Di is no longer needed.

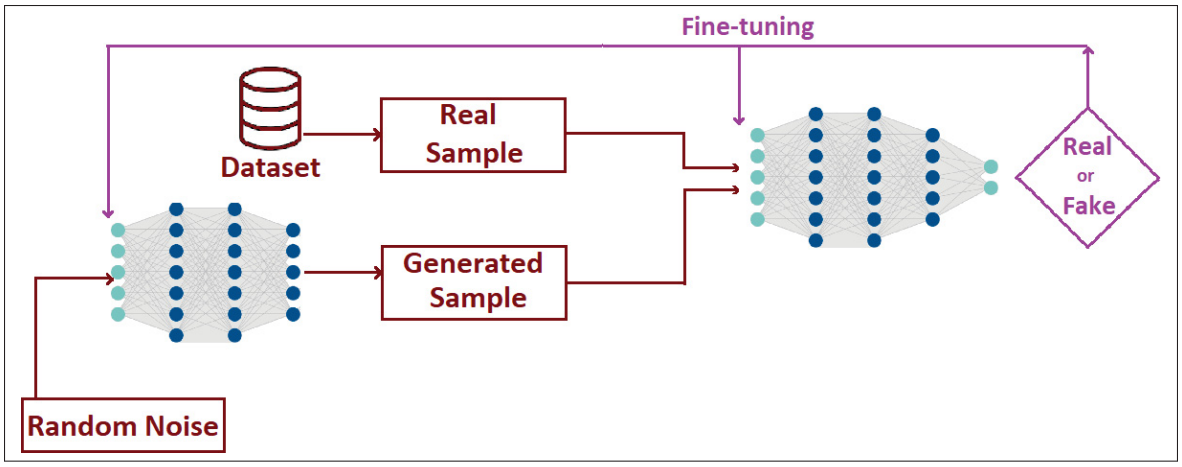

Figure 1.2 GAN training process: The generator creates synthetic samples from random noise, attempting to mimic real samples from the dataset. The discriminator evaluates these generated samples alongside real samples, providing feedback on their authenticity. This feedback is used to fine-tune the generator's neural network, progressively enhancing the quality of the generated samples

To learn the generator's distribution, denoted by $p_g(x)$, over data samples (x), I first define a prior distribution on the input noise variables, $p_z(z)$, which is used as the input of Ge. Ge learns a mapping from noise (z) to data space, $G(z;\theta_g)$ with the parameter set θ_g. Additionally, the discriminator $D(x;\theta_d)$, which outputs a single scalar value, $D(x)$, represents the probability that the input data x originated from the real data distribution rather than the generator's distribution (p_g). The goal of training Di is to maximize the probability of assigning the correct label to both the generated and real samples (Goodfellow *et al.* (2014)). Ge is trained simultaneously to minimize $\log(1 - D(G(z)))$. In other words, Di and Ge play the following minimax game involving finding the optimal parameters of θ_g and θ_d:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]. \tag{1.1}$$

The result of this adversarial training is a generator capable of producing high-quality data, and a discriminator unable to differentiate real samples from generated samples.

1.1.3 GAN for improving performance of DRL

In this section I consider related works about application of GAN to improve the performance of DRL for 5G resource allocation.

In (Kasgari *et al.* (2020)), GAN is used to augment training dataset for DRL in order to improve the reliability of DRL encountering rare situation in real world. Authors proposed a framework for model-free downlink resource allocation of URLLC which guarantees high reliability and low latency. They proposed a new CGAN architecture able to generates extreme environment situations such as sudden traffic increase or sudden channel gain decreases. Using these augmented dataset in a simulator to train DRLs, give them enough experience to deal with them in real world which increases the URLLC reliability by optimal OFDMA resource allocation.

To enhance the effectiveness of DRL in 5G network slicing, (Hua, Li, Zhao, Chen & Zhang (2019)) employed a Wasserstein GAN to estimate the action-value distribution of the DRL agent, thus implementing a distributional RL approach. The presence of random noise in the environment often leads to either overestimation or underestimation of action values in value-based RL algorithms like DQN. By adopting distributional RL for resource allocation across slices, the performance of 5G networks is improved, consequently enhancing the reliability of slices for providing URLLC service.

Although both of the aforementioned works used GANs to enhance the performance of RL, their proposed models do not account for the computational constraints present at the Base Station and edge devices. A 5G network environment is dynamic and undergoes frequent changes, including factors such as network traffic, user density, and signal gain and interference levels. Therefore, the training process of GANs and other DNN models must occur rapidly enough to adapt to these environmental changes. Otherwise, the datasets generated by the GANs risk becoming outdated, rendering them ineffective for training other DNN models. Since the computational power at the Base Station and edge devices is typically limited compared to centralized data centers, this limitation can lead to delays in training the GANs, making it difficult to keep up

with the fast-paced changes in the network environment. Consequently, the GAN-generated data may not reflect the current state of the network, leading to suboptimal performance of the RL models trained on this data.

1.2 Related works

In general, related works either focus on improving the reliability of DRLs or use GANs to enhance the performance of DRLs. Apart from data augmentation techniques, there are numerous other methods, such as transfer learning and meta-learning. In transfer learning, knowledge from a trained DRL agent is transferred to a target agent. In 5G networks, transfer learning has shown performance improvements. For example, in (Zhou, Erol-Kantarci & Poor (2022)), it is used to enhance the performance of distributed DRL agents for resource allocation.

However, transfer learning in wireless communication and dynamic environments, such as those involving DRL agents, presents several challenges. These include the risk of negative transfer, where knowledge from another domain could degrade performance rather than improve it. Additionally, in dynamic wireless environments, frequent updates to DNN weights to adapt to changes can erase or render outdated any transferred knowledge. Furthermore, finding a source domain sufficiently similar to the target domain is difficult, which is essential for successful transfer learning.

Moreover, meta-learning is another approach that aims to train models capable of quickly adapting to new tasks with minimal data (Eghbali, Taskou, Mili, Rasti & Hossain (2024)). One of the challenges faced with meta-learning is determining the optimal curriculum for training, as well as adapting meta-learning algorithms to diverse environments. Meta-learning also requires substantial computational resources and careful tuning to ensure the model can generalize across various tasks and conditions.

The remarkable ability of GANs to produce diverse and realistic data has led to the development of new applications, particularly in the areas of vision and audio. These advancements are the result of modifications to the vanilla GAN framework, including innovative loss functions for

training GAN such as Wasserstein GAN (Arjovsky, Chintala & Bottou (2017)), diverse DNN architectures such as deep convolutional GAN (DCGAN) (Radford, Metz & Chintala (2015)), the incorporation of additional generators or discriminators (Hardy, Le Merrer & Sericola (2019)), and hybrid approaches that combine GANs with other generative models like variational autoencoders (Khan, Hayat & Barnes (2018)). These innovations have led to exciting applications such as handwritten font generation and image inpainting, text-to-image synthesis, human pose synthesis, image manipulation applications, object detection, three-dimensional (3D) image synthesis, image super-resolution, image-to-image translation, language and speech synthesis, music generation, video applications etc. (Gui *et al.* (2021)).

The capability of GANs to produce high-quality synthetic data presents a significant advantage within 5G networks, especially in scenarios where acquiring large datasets is costly or unfeasible. Prior research has leveraged GANs to learn data distributions, addressing challenges such as data scarcity or the creation of new datasets tailored for 5G applications. This subsection examines earlier studies focused on utilizing GANs within the context of 5G networks. I categorize related works into different sections, each emphasizing the application of GAN in 5G networks from various perspectives.

1.2.1 Training GAN

In this section I consider previous related works used GAN for data augmentation in 5G networks.

A framework is proposed for training Conditional Generative Adversarial Networks (CGANs) in unmanned aerial vehicle (UAV) wireless networks (Zhang, Ferdowsi, Saad & Bennis (2021b)). In their proposed framework, each UAV (act as a BS) trains a CGAN using a limited number of channel samples from each of their visited locations to be shared with other UAVs. Their work considers the communication overhead for data exchange among SBSs for distributed training of multiple GANs. However, it does not account for the training time, channel distribution change, and computational resources required for the training process.

In another related work, a framework for privacy-aware task offloading has been developed as proposed by (Xu *et al.* (2020)) in which trains multiple distributed GANs concurrently. Like my proposed method, they train multiple smaller GANs, each for a user considering overall network performance metrics such as computation load balancing and reducing dataset transmission time. However, their main objective is to preserve the privacy of users data tailored for training GANs in domains such as image generation. But, in 5G networks optimization, networks data such as channel gain etc. are less sensitive to privacy leakage. Moreover, in 5G URLLC environments, particularly in harsh industrial settings, the dataset distribution (environment) changes more frequently, with shifts occurring on the order of minutes. As a result, the frequency of GAN training must be higher, and the training deadlines are tighter. With respect to the MEC computation resource constraints such a training requires huge computation power on edge that makes application of their proposed approach for my case challenging.

A framework called collaborative game parallel learning is proposed by Zhang et al. to train GANs distributed using federated learning to preserve users' privacy (Zhang *et al.* (2023)). Similar to my work, it addresses communication constraints, and limited computation in geographically distributed nodes for GAN training. However, unlike my work that trains a dedicated GAN for each user in cloud, they train a single general purpose GAN to solve the problem of data scarcity of nodes. The main problem with their approach is that when there is little similarity between different nodes, the probability of GAN training divergence is high. To solve this problem I train several distributed GANs and also decrease the computation and time for training process by sending training dataset to cloud.

1.2.2 Resource Allocation for Training DRLs

In their work, (Jia, Zhou, Xu & Jin (2021)) propose a framework for optimal continuous training of DNN models in IoT environments. This framework is comparable with my work if I consider training DRL as DNN models. Their proposed framework focuses on the optimal allocation of resources to reduce the training costs in resource-constrained Multi-access Edge and cloud. Their model also considered environment change by monitoring performance of DNNs while in

my model I consider the environment as episodes. However, other factors such as training time and scheduling training of multiple DRLs is overlooked. Moreover, their model trains DNNs in distributed nodes in edge and cloud while I train a GAN and DRL in a single nodes.

1.2.3 Discussion

In this chapter, I provide an overview of relevant studies, as shown in Table 1.1, and compare their characteristics to my research. The fundamental difference is that my approach considers communication and computation resources, training time, and the scheduling of multiple GANs simultaneously, with the objective of increasing reliability, while previous works, as shown in Table 1.1, only consider a subset of these parameters or train GANs with different objectives

Table 1.1 Literature review of the related works

Research work	DRL/DNN	Training Time	URLLC	Multiple GANs	Training Objective	Communication	Computation
(Kasgari *et al.* (2020))	DRL	—	✓	—	Reliability	—	—
(Xu *et al.* (2020))	—	—	—	✓	Privacy & Load balancing	✓	✓
(Zhang *et al.* (2023))	—	—	—	—	Privacy & Federated Learning	✓	✓
(Hua *et al.* (2019))	DRL	—	✓	—	DRL output quality	—	—
(Jia *et al.* (2021))	DNN	✓	—	—	Quality & Training Time	✓	✓
(Zhang *et al.* (2021b))	—	—	—	✓	Time & Energy	✓	—
OGAN (My solution)	DRL	✓	✓	✓	Reliability	✓	✓

CHAPTER 2

METHODOLOGY

In this chapter I present my architecture as well as the optimization model for GAN, called OGAN. First I present the system model and four components of my architecture as in Figure 2.1, Small Base Station, Macro Base Station. Finally, I present my optimization model, called OGAN followed by a DC-algorithm to solve it.

2.1 System Model

Due to the high cost, geographic deployment challenges, and the need for system agility, it is impractical to forward the backhaul traffic of every SBS via dedicated broadband Internet or fiber links, particularly in urban environments. I assume that the backhaul traffic of SBSs is forwarded to the core network through a shared wireless link between the SBSs and a MBS. Both the MBS and SBS can independently transmit user data and management data to their associated users. Although users can hand over between SBSs within macrocells, for simplification, I assume that during each wireless episode, users are served by a single SBS. For efficiency, the gateway is configured at the MBS, which typically has sufficient space to install massive antennas to receive the wireless backhaul traffic from SBSs. However, the capacity of the backhaul network remains a bottleneck, potentially limiting the densification of small cells in 5G UDNs (Ge *et al.* (2016)).

The downlink channel of each SBS is shared based on OFDMA to deliver URLLC service. UDN network serves a set $\mathcal{L}$ of L users distributed evenly among all SBSs. For simplicity, I assume each user l is only served with a single SBS i, $\forall i \in \mathcal{N}$ and $\forall l \in \mathcal{L}$. The downlink transmission rate from the SBS i to user l is given by:

$$r_l(t) = \sum_{z=1}^{K_i} \rho_{lz}(t) B \log_2 \left(1 + \frac{p_{lz}(t) h_{lz}(t)}{\sigma^2}\right), \tag{2.1}$$

where B is the bandwidth of the resource block (RB), and $h_{ilz}(t)$ represents the time-varying channel gain of the transmission from the SBS i to user l on RB z at time slot t. $p_{ilz}(t)$ denotes

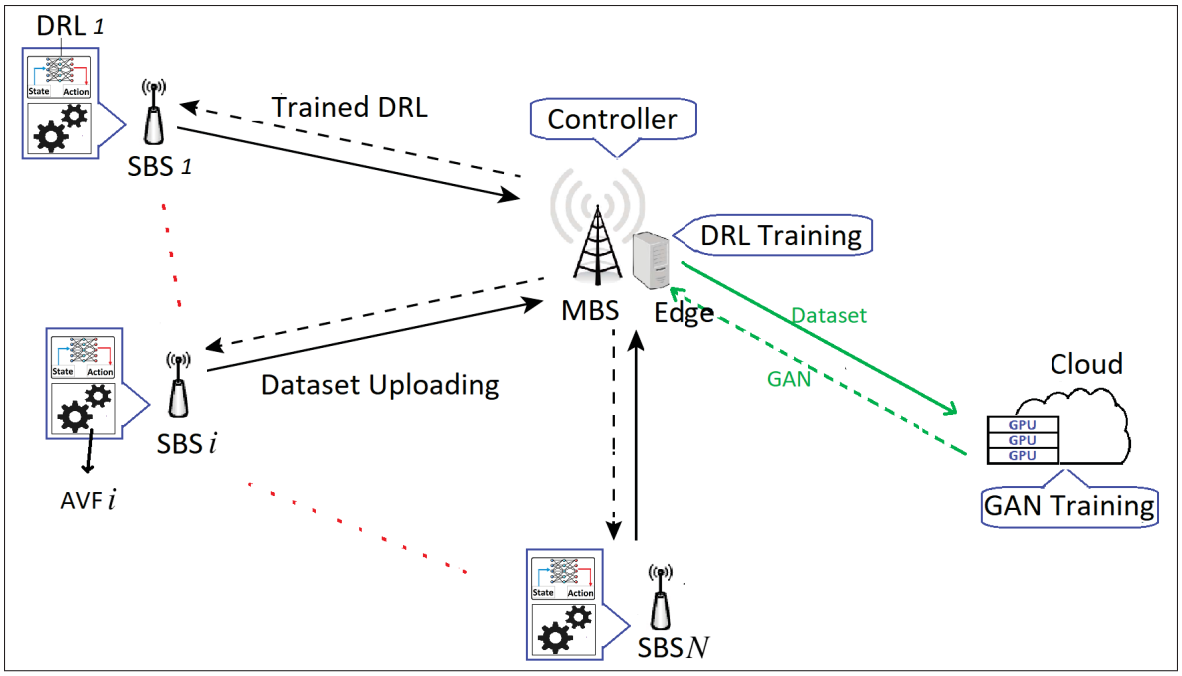

Figure 2.1 System Model. A 5G UDN with a single MBS and N SBSs. The architecture shows the workflow for training DRL models at the edge and GANs in the cloud. SBSs collect and upload GAN training datasets to the MBS, which then forwards them to cloud.

the downlink transmission power of the BS i on RB z to user l at slot t. K_i is the number of RBs of SBS i while $\rho_{lz}(t)$ serves as the RB allocation indicator, with $\rho_{ilz}(t) = 1$ when RB z is allocated to user l from SBS i. OFDMA resource allocation is performed distributed on each SBSs i with its dedicated DRL agent, DRL i where $i \in \mathcal{N}$. Following I present formulation of the DRL agent.

2.1.1 Deep Reinforcement Learning

I consider the DRL model presented in (Kasgari *et al.* (2020)) for 5G URLLC OFDMA resource allocation, as shown in Figure 2.2. At each transmission interval, DRL i receives the current environment state and reward from the SBS i as input. The DRL model incorporates two inputs as its reward to evaluate its performance and update its DNN parameters in each time slot: the total downlink power of SBS i denoted as $P_i(t) = \sum_l \sum_z p_{lz}(t)$, and the measured reliability of

each user at each time slot t is $(1 - \gamma_l(t))$ where $\gamma_l(t)$ is given by:

$$\gamma_l(t) \approx 1 - \frac{\mu_l^{'}(t)}{\mu_l(t)}, \tag{2.2}$$

where $\mu_l^{'}(t)$ represents the number of packets received by user l and $\mu_l(t)$ is the total number of packets transmitted to user l at time slot t. Note that it is an empirical measurement for unreliability so there is no requirements for assumptions about unreliability.

The channel gains of each user, denoted by $h_{lz}(t)$, the number of packets $\hat{\mu}_l(t)$ transmitted to each user l, and the average packet length $\hat{\beta}_l(t)$ for each user l are the environment state of agent, formulated as $s_t = (\hat{\mu}_l(t), \hat{\beta}_l(t), h_{lz}(t))$ for all $l \in L$, $z \in K$.

Using these parameters as the state space, the DRL agent learns optimal policies for OFDMA resource allocation. To do that, DRL i determines $\rho_{lz}(t)$ and $p_{lz}(t)$, its action, for all users served by SBS i. Through iterative assignment of $\rho_{lz}(t)$ and $p_{lz}(t)$ and receiving feedback across multiple time slots, the system can consistently uphold user reliability, latency, and throughput. The action $a_t = (p_{lz}(t), \rho_{lz}(t))$ for all users l and RBs z which are the transmission power and RB allocation to users. Subsequently, the reward of DRL is determined to to ensure the QoS for URLLC.

Since SBSs have different characteristics, such as channel distributions, traffic patterns (Zhang, Dang, Shihada & Alouini (2021a)), it is not feasible to implement a single DRL model for all SBSs. Therefore, each DRL i is trained based on the dataset gathered from SBS i to improve their performance. Although it is possible to train a single larger DRL model to be used for all SBSs, it increases the DNN size of DRL agents since they should learn data distribution of all of SBSs. By increasing DNN size, the training and inference time of DRL as well as computation and energy consumption of SBSs to take actions is increased. I then use a smaller DNN models for DRL and train a dedicated DRL based on dataset gathered from its SBSs. This way, with smaller DRL models, less computation is required for inference of DRL model that decreases the size, price and battery usage of SBSs.

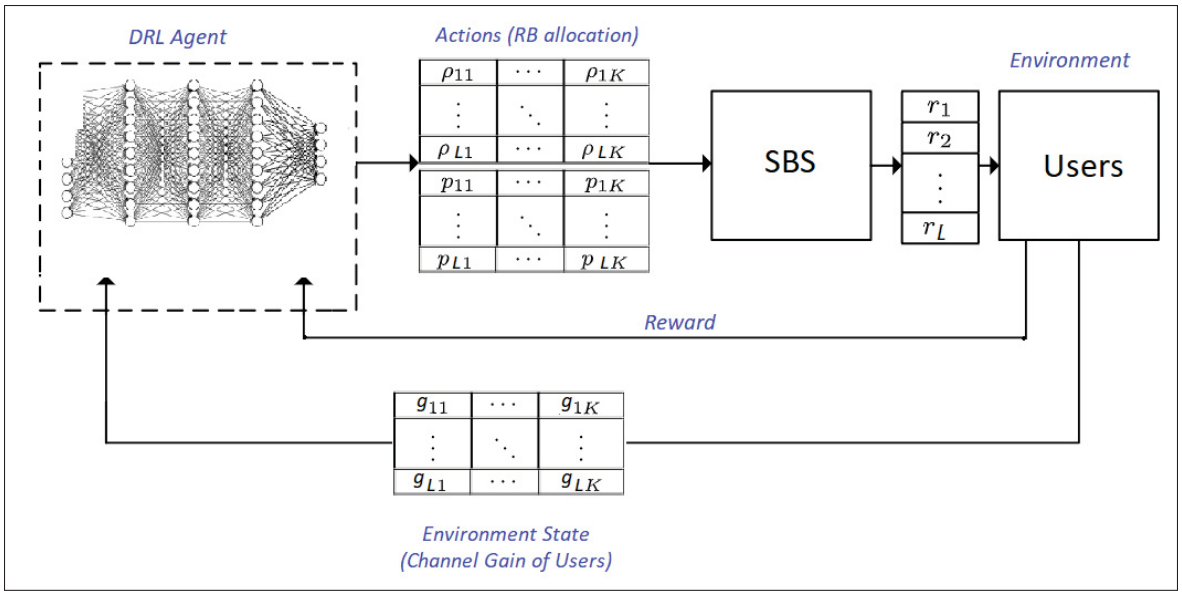

Figure 2.2 DRL model for OFDMA resource block (RB) allocation in a cell with L users and K sub-carriers. The DRL agent takes the environment state (Channel Gain Matrix of users) and outputs actions in the form of RB allocation matrices. These actions are then used by the SBS to determine the transmission rate of users. The reward is fed back to the DRL agent to update its policy and improve future allocations.

I consider a dynamic network scenario where changes occur slowly over time and are divided into episodes. I assume in each episode, the channel and traffic distribution of SBSs remain constant, as in (Sun & et al. (2022)). For SBS i, the remaining time till the end of the current episode is given as T_i^{max}. Episode time is based on application and other parameters which affect the channel distribution. Following I propose some approaches to estimate the value of T_i^{max}.

In the industrial environment, numerous factors can significantly impact channel distribution. Even slight alterations in these factors can be seen as significant channel distribution shifts. Challenges stemming from dust, temperature fluctuations, vibrations, humidity levels, object trajectories, materials, obstacles, and lighting are more pronounced in industrial settings compared to other environments. Additionally, signal interference from wireless devices and motors further complicates wireless communication in such environments. Given that many of these factors are linked to industrial processes, one approach to determining T_i^{max} is to forecast

it based on the factory's production schedule, taking the time of process changes as T_i^{max} (Li *et al.* (2017)).

In open spaces, weather forecasts can be utilized to predict variables such as temperature, humidity, precipitation, and sunlight, aiding in the determination of T_i^{max}. Similarly, in urban environments, in addition to the aforementioned factors, analyzing traffic patterns can provide valuable insights for predicting T_i^{max}.

Another general approach to determining T_i^{max} involves prediction methods that estimate the number of active users or cell traffic load (Wang *et al.* (2022b)). Sudden changes in these factors can alter the state-space distribution of DRL, potentially leading to the conclusion of the current wireless episode.

Additionally, another way for UDNs to predict T_i^{max} is by forecasting SBS on/off patterns (Zhu & Wang (2021)). In UDNs, frequency reuse indicates the level of interference from neighboring cells. When an SBS is toggled on or off, it affects the interference level of adjacent cells, thereby altering the channel distribution, which constitutes the state-space of a cell. Therefore, predicting the on/off status of cells can serve as T_i^{max} for other adjacent cells using the same frequencies.

2.1.2 GAN model

The presence of rare samples within the state-space of DRL can lead to unreliable URLLC service. To mitigate this issue, I train a GAN with rare samples to augment the training datasets for each DRL. Given my assumption that the state-space distributions of DRLs from different SBSs are dissimilar, it follows that the distributions of their rare samples also differ. Consequently, I propose for training a dedicated GAN for each SBS, which reduces the time and computational resources required for training. Training multiple smaller GANs consumes less time and allows for parallel training across multiple GPUs, thereby enhancing training speed. Subsequently, each GAN is utilized to augment the training dataset for its corresponding DRL. Therefore, for each SBS i, I propose training a GAN, denoted as GAN i. The training dataset for

GAN i comprises rare samples collected from SBS i, enabling more effective augmentation of the training datasets for DRL i.

2.1.3 GAN

In this section I present the GAN model used for augmenting rare smaples. My GAN has two DNNs, a Generator and a Discriminator, which are trained concurrently through adversarial processes. The architecture and functionality of each component are detailed below.

2.1.3.1 Discriminator

The Discriminator is a convolutional neural network that classifies input (OFDMA channel gain matrix) as real or generated (fake). Its output is a single scalar value indicating the probability that the input is real. The Discriminator network structure includes:

1. **Convolutional Layers**: The Discriminator consists of five convolutional layers. The first layer takes a single-channel input image and applies a convolutional filter to produce feature maps. Subsequent layers increase the number of feature maps while reducing the spatial dimensions, allowing the network to capture increasingly abstract features.
2. **Activation Functions**: Leaky ReLU activations with a negative slope of 0.2 are used after each convolutional layer, except the final layer, where a sigmoid function is applied to produce a binary classification output.

2.1.3.2 Generator

The Generator is a deconvolutional neural network (also known as a transposed convolutional network) that synthesizes fake samples (channel gain matrix) from random noise vectors. The architecture of the Generator includes:

1. **Transposed Convolutional Layers**: The Generator consists of five transposed convolutional layers. These layers progressively upsample the input noise vector to produce a high-resolution output. The first layer transforms a 100-dimensional noise vector into feature

maps with depth '8D' and applies batch normalization. The final (fifth) layer outputs a single-channel matrix with parameter values in the range $[-1, 1]$.

2. **Activation Functions**: ReLU activation is used after each transposed convolutional layer except the final layer, where a tanh function is applied to ensure the output image has pixel values in the desired range.

2.1.3.3 Training Procedure

The training process involves alternating updates to the Generator and the Discriminator:

- **Discriminator Training**: The Discriminator is trained to maximize the probability of correctly classifying real samples and minimize the probability of incorrectly classifying generated samples.
- **Generator Training**: The Generator is trained to minimize the probability of the Discriminator correctly identifying generated fake samples, effectively maximizing the Discriminator's error rate.

The overall objective is to reach a Nash equilibrium where the Discriminator cannot distinguish between real and generated samples better than random chance, thus ensuring the Generator produces highly realistic (rare) samples.

2.1.4 Architecture for training GAN/DRL in UDN

In this section I explain my architecture for optimal resource allocation for training GAN and DRL in resource limited 5G UDNs. My proposed architecture is consisted of four main elements, small base station (SBS), macro base station (MBS), mobile edge computing (MEC) and cloud.

Due to the computational limitations of SBSs, DRL models and GANs are trained on a Mobile Edge Computing (MEC) server associated with the MBS or on the cloud data center. SBSs often lack the processing power and memory required for intensive training tasks, which necessitates offloading these tasks to more capable servers.

Training GANs, in particular, is computationally intensive due to their DNNs with a large number of parameters, resulting significantly longer training times when using CPU servers. In contrast, training DRLs in a simulator, although less demanding than GANs, still require substantial computational resources, mostly CPUs since simulator environment demands more CPU powers. If the training time exceeds a predefined maximum duration, wireless episode (T_i^{max}) in my case, both the GAN and DRL models can become outdated. This is because the environment's dynamics and the data distribution used for training these models may change significantly after T_i^{max}, rendering the models less effective or even obsolete.

To address these issues, my architecture proposes offloading training GANs in the cloud, where powerful GPUs are optimized for accelerating the training process of GANs. This setup significantly reduces the training time, ensuring that the models are updated before T_i^{max} is reached. The cloud infrastructure also provides scalable resources, allowing for the training of larger and more complex models as needed.

Additionally, the MEC server plays a crucial role in maintaining the performance of DRL models. By offloading the less computationally demanding DRL training tasks to the MEC server, we can achieve a balance between latency and computational efficiency. This setup enables real-time decision-making for the SBSs while still benefiting from the robust training capabilities of the cloud for GANs.

In summary, my architecture leverages the strengths of both MEC servers and cloud data centers to address the computational challenges of training DRLs and GANs. By strategically offloading tasks based on their computational requirements, we ensure timely updates and maintain the effectiveness of the models in dynamic environments.

Another rationale for training GANs in the cloud is the relatively smaller dataset size required for GAN training compared to training DRL models. In my problem, GANs typically utilize datasets consisting of only a few thousand samples, whereas DRL models require datasets with tens of thousands of samples. Consequently, training GANs in the cloud generates less traffic on the connection link between MBS and the cloud infrastructure compared to training DRL

models in the cloud. By offloading GAN training to the cloud, I minimize the burden on the connection link, ensuring efficient data transfer without overwhelming the network.

On the other hand, I use servers with CPUs on edge for training DRLs. Since training DRL with real data samples gathered from cells causes traffic challenge, we have to store those datasets on MEC and use MEC resources to train DRLs. It has two other benefits also: Firstly, as mentioned before DRLs model used for OFDMA resource allocation is much smaller than GAN model. So its training on CPU servers take less time compared to training GAN on the same server. Moreover, during training of DRL in a simulator, more demands CPU resources than GPUs.

On the contrary, we employ servers equipped with CPUs at MEC for training DRL models. Training DRL models with real data samples collected from SBSs presents a traffic challenge, necessitating the storage of these datasets on MEC platforms. By utilizing MEC resources for DRL training, we not only mitigate traffic challenges but also capitalize on two additional benefits. Firstly, as previously mentioned, DRL models utilized for Orthogonal Frequency Division Multiple Access (OFDMA) resource allocation are significantly smaller compared to GAN models. Consequently, their training on CPU servers consumes less time in comparison to training GANs on the same server. Furthermore, during the training of DRL models in a simulator, there is a higher demand for CPU resources than for GPUs. By leveraging CPU resources at the edge for DRL training, I optimize computational efficiency while also addressing traffic challenges associated with real data sample usage. This approach ensures timely and effective training of DRL models, facilitating their deployment in communication systems with minimal disruption.

Following I present the detail of four components of my architecture which is controlled with a central controller.

2.1.4.1 Small Base Station

Here, I will explain the first element of my architecture. The main responsibility of the SBS is to provide URLLC service to users within its coverage area. It employs a DRL agent for

optimal OFDMA resource allocation. Another responsibility of the SBS is to collect datasets for training DRL and GANs, sending these datasets via the backhaul to the MEC or cloud through the MBS. Additionally, the SBS has an AVF agent, discussed in the next section, to collect and store rare samples from the environment for GAN training. As previously mentioned, to remain cost-efficient, SBSs have limited computational power, insufficient for training DNNs. Therefore, the training of DRL and GANs should be conducted on the MEC and/or cloud.

Following this, I will provide a more detailed explanation of the components of the SBSs.

2.1.4.2 Finding Rare Samples

To detect rare state-spaces, I utilize a failure probability predictor known as AVF (Uesato & et al.). The primary role of the AVF agent is to identify failure situations encountered by a DRL agent. Typically, Monte Carlo simulations are employed to find failure situations and estimate their probabilities by comprehensively testing various conditions. However, AVF leverages data already collected during the training and implementation processes, eliminating the need for extensive Monte Carlo simulations to identify failure conditions. This approach streamlines the identification of rare state-spaces and enhances the efficiency of failure detection in DRL systems.

Given that packet loss significantly impacts URLLC reliability, I designate it as a critical failure scenario for my AVF model. The objective of AVF is to find cases where the DRL agent may fail to provide reliable communication. The episode duration for AVF is determined by $\min(d_0, ..., d_i, ..., d_N)$, where d_i represents the URLLC transmission delay of SBS i. In each SBS i, an AVF agent, AVF i, continuously monitors the DRL agent's performance to identify failure cases to be used as rare samples. The AVF agent then maintains a dataset Υ_i, which contains these rare samples for training GAN i.

The dataset Υ_i accommodates N_{Υ_i} samples, capturing instances of failure scenarios. When Υ_i reaches its capacity or when a wireless episode transitions, the oldest samples are replaced by

the most recent ones. This ensures that the dataset remains up-to-date and relevant, continuously providing valuable data for training GANs to improve system robustness.

2.1.4.3 Generative Adversarial Network

During the training of a GAN, its discriminator (Di) evaluates the quality of the generated samples produced by the generator (Ge) using feedback measures known as Neural Net Distance (NND). NND quantifies the degree of dissimilarity between the generated data and the real dataset. Various metrics can be employed as NND during the training of GANs. The training process involves solving a minimax problem through empirical risk minimization, wherein an estimation error ϵ is associated with NND. Estimation error ϵ refers to the difference between the empirical risk (i.e., the average loss over the training data) and the true risk (i.e., the expected loss over the entire data distribution).

Based on the analysis of Rademacher complexity conducted by the authors in (Ji, Zhou & Liang (2021)), they identify an upper bound error associated with NND, denoted as E. It's noteworthy that their investigation focuses on GAN models employing Rectified Linear Unit (ReLU) activation functions. In their proposed model, they consider the size of the training dataset, denoted as N_Υ, and the number of unique fake samples, denoted as m, generated and evaluated during the training phase. Particularly, when the dimension of the input noise vector is large, such as 128, the parameter m can be regarded as the number of training iterations for the GAN model. Inspired by their research, I suggest a compressed version of E as follows:

$$E = \frac{V}{\sqrt{m}} + \frac{W}{\sqrt{N_\Upsilon}}, \tag{2.3}$$

where V and W are sums of the Frobenius norm of the generator and discriminator parameter matrices, respectively. When E reduces, GAN's probability distribution of the generated data samples approaches the probability distribution of real data. Note that Eq. 2.3 indicates that by increasing the values of m (number of GAN training iterations) and N_Υ (number of training

samples), the value of E decreases. This observation implies that as more training iterations and samples are utilized, the upper bound error associated with NND in GAN models diminishes.

According to (Zhou *et al.* (2020)), V and W can be obtained through offline training of GANs using different sample size N_Υ. The sizes of my datasets used for offline training are selected from the list $(N_\Upsilon^{(1)}, N_\Upsilon^{(2)}, N_\Upsilon^{(3)}, N_\Upsilon^{(4)})$ = (1,000, 2,000, 3,000, 4,000). After conducting 500 GANs trials for each values of N_Υ, I calculate the averaged values for V and W, labeled as $\bar{V}_{N_\Upsilon}$ and $\bar{W}_{N_\Upsilon}$, respectively. The resulting tuples of $\{\bar{V}_{N_\Upsilon}, \bar{W}_{N_\Upsilon}\}$ corresponding to the aforementioned list are given by ({14.41, 13.17}, {15.2, 14.1}, {14.80,13.78}, {14.52, 13.42}).

Table 2.1 Offline Training Results

N_Υ	$\bar{V}_{N_\Upsilon}$	$\bar{W}_{N_\Upsilon}$
1,000	14.41	13.17
2,000	15.2	14.1
3,000	14.80	13.78
4,000	14.52	13.42

Resolving the critical challenges of mode collapse and divergence in GAN training typically demands larger datasets (Bang & Shim (2021)). In my case, a minimum of 1,000 samples is required to train GANs to achieve the acceptable quality, based on my experimental findings. When the dataset is large and complex, training GANs in resource limited environments, is more challenging since in high dimensional sample spaces, GANs struggles to cover all modes. Hence, for each SBS, I train multiple smaller GANs to exploit the temporal correlations among data samples collected in close proximity. This strategy not only reduces the training time and memory requirements of GANs, but also reduces the risk of mode collapse and training divergence.

2.1.4.4 Cloud Datacenter

As mentioned before, in 5G URLLC environments, the dataset distribution (environment) changes more frequently, with shifts may occur on the order of minutes. As a result, the frequency of GAN training must be higher, and the training deadlines are tighter. To reduce

training time, it is required to train them in powerful GPUs provided by cloud datacnters. Furthermore, I train multiple smaller GANs (one for each SBS) rather than a single larger GAN shared among all SBSs. This strategy reduces the dataset size significantly compared to training large GANs for high-quality image or video generation. Consequently, the transmission time required for sending the dataset to the cloud is reduced, enabling us to leverage the higher processing power of GPUs available in the cloud, rather than relying on weaker MEC servers.

The cloud datacenter is equipped with N_g GPUs, each operating at a frequency of F_g. For simplicity, I assume that each GAN is trained on a single GPU, and GPUs are only released once the training task for their respective GAN is completed. This straightforward setup simplifies the resource allocation process and ensures that each GPU is fully dedicated to the training of its assigned GAN until completion. SBSs upload their datasets, denoted as Υ_i, to the cloud through the MBS. The training time of GAN i is given by $T_i^{gan} = m_i.\eta_i^{(g)}/F_g$, where m_i is the number of training steps for GAN i ($\mathbf{m} = \{m_i\}$), $\eta_i^{(g)}$ denotes the required GPU cycles for GAN i, and F_g is the processing capacity of the GPU. Finally, the trained GANs are transmitted back to the MEC platform for dataset augmentation.

2.1.4.5 Mobile Edge Computing (MEC)

The large datasets used for DRL fine-tuning should be hosted in the MEC to reduce data transmission overhead to the cloud. Since training a DRL requires very large datasets, in my architecture I propose to train DRLs in the MEC to decrease the overhead of sending these datasets to the cloud. I assume that a dataset from the current episode is available in the MEC. Each time we train a new GAN, we add the augmented dataset from the GAN to this existing dataset.

The MEC has N_c CPU cores in total, each operating at a frequency of F_c GHz. The pre-training time for DRL i is given by $T_i^{tr} = (\Phi_i.\eta_i^{(c)})/(F_c.c_i)$, where c_i is the allocated number of CPU cores, and Φ_i is the number of steps to train DRL i. I denote the vector $\mathbf{c} = \{c_i\} \in \mathbb{R}^N$. A maximum of Ψ_i CPU cores can be used for training DRL i. Since the data distribution remains

constant within an episode, the number of training iterations for DRL agent i, denoted as Φ_i, is estimated based on the number of training iterations of its current version. As all DRL agents are pre-trained, Φ_i is readily available from the start.

The time required for SBS i to upload dataset Υ_i to MBS is $T_i^{com} = |\Upsilon_i|/R_i$, where $|\Upsilon_i|$ is the size of Υ_i in MByte, and R_i is the available bandwidth from SBS i to the MBS given by $R_i = B_i.log_2(1 + SNR_i)$ where B_i is the channel bandwidth of SBS i and SNR_i is its average signal-to-noise ratio which is assumed to be constant over a long period of time (Xie, Xia, Wu, Wang & Poor (2023)). Besides, since the size of DNNs (GAN and DRL) is very small compared to the dataset, their transmission time can be discarded or be a small constant number in my model.

2.1.4.6 Central Controller

The central controller located on the MBS, is responsible for monitoring and allocating system resources by prefetching commands from other components. Once a GPU or CPU core is released, the controller runs an optimization model named OGAN (see Section 2.2), to schedule GAN and DRL training. OGAN determines the SBSs to train GANs as well as the optimal training steps **m** and computational resource **c**. The controller's routine, as shown briefly in Algorithm 1, involves the following main steps:

1. Continuously monitor both MEC and cloud to detect changes in available GPUs (N_g) and CPU cores (N_c).
2. If N_g or N_c changes and both of them are greater than zero, initiate pre-fetching of the following parameters from SBS i: ν_i^t (total transmitted packets), ν_i^l (total packet loss), $|\Upsilon_i|$ (dataset size), and B_i (available back-haul bandwidth).
3. Calculate a weight factor $\beta_i = \nu_i^l/\nu_i^t$ to assess the reliability of SBS i. For a fair comparison of SBSs based on their contribution to system unreliability, compute the normalized $\tilde{\beta}_i$.
4. Solve OGAN to determine whether a GAN should be trained for SBS i (variable **z**), then allocates resources on cloud (variable **m**) and MEC (variable **z**) accordingly.
5. If **z** == 0, return to the initial step, awaiting to change in N_c and N_g.

Algorithm 2.1 Controller Algorithm

Input: inputs set:$(\eta_i^{(g)}, \eta_j^{(c)}, E_i^{min}, \Psi_i, T_i^{max}, \Phi_i, B, B_i, N_\Upsilon, N_g, N_c, N, F_g, F_c)$
Output: outputs set:(m_i, c_i, z_i)

1 Monitor to detect changes in N_g and N_c
2 **for** *(all GPUs and CPUs)* **do**
3 **if** *(changes in N_g or N_c occur)* **then**
4 **if** *($N_g > 0$ and $N_c > 0$)* **then**
5 **for** *(all SBS $i \in \mathcal{N}$ $(i = 1, ..., N)$)* **do**
6 Pre-fetch $\{v_i^t, v_i^l, |\Upsilon_i|, B_i\}$
7 Calculate $\beta_i = \frac{v_i^l}{v_i^t}$
8 **end for**
9 Solve OGAN
10 **for** *(all SBS $i \in \mathcal{N}$ $(i = 1, ..., N)$)* **do**
11 **if** *($z_i == 1$)* **then**
12 send **c** to MEC
13 send **m** to cloud
14 send z_i to SBS i to upload GAN training dataset
15 **end if**
16 Initiate GAN training
17 After GAN training completion, send every GAN i to SBS i
18 Initiate DRL training
19 Implement trained DRL i on SBS i
20 **end for**
21 **end if**
22 **end if**
23 **end for**
24 return to the initial stage (line 1)

6. When the training is compeleted, send GAN generator DNNs to the MEC, where they are employed to augment the training datasets for their respective DRLs.
7. Start training (fine-tuning) new DRLs at the MEC servers, and when they are completed, replace the existing ones with the newly trained ones.

Fine-tuning DRL i is performed with c_i CPU cores. Each time OGAN is called, it may dynamically adjust the number of cores assigned to a DRL during its training at the MEC, which might involve decreasing c_i to increase c_j, $\forall i, j \in \mathcal{N}$. As a result, this algorithm maximizes

the efficiency of both cloud and MEC in order to train GAN and DRLs, ultimately resulting in higher quality and reliability for URLLC.

Note that, since the number of SBSs exceeds the number of available GPUs, it is impossible to train all SBSs in one round. The maximum number of SBSs selected for training each time OGAN is called is equal to the number of GPUs, N_g. Whenever the training of a GAN model on a GPU is completed, or when the training of one of the DRL models finishes and CPU cores are released, OGAN is called again to determine resource allocation and assign resources for training the current GANs or choosing a new one.

Table 2.2 OGAN's input Parameters

Input Parameters	
$\eta_i^{(g)}$	The number of GPU cycles used in each iteration of GAN i training
$\eta_j^{(c)}$	The number of CPU cycles used for each step in pre-training DRL i
E_i^{min}	Lower bound of the estimation error for GAN i
Ψ_i	Maximum number of CPU cores can assigned for pre-training of DRL i
T_i^{max}	Remaining time until the end of current wireless episode
Φ_i	The upper bound of the training step for pre-training of DRL i
B	Shared bandwidth from SBS i to MBS
B_i	Bandwidth from SBS i to MBS
N_Υ	Dataset storing rare samples of DRL i
N_g	The number of GPUs available in the cloud
N_c	The number of CPU cores available in the edge
N	Number of all SBSs
F_g	Frequency of each GPU
F_c	Frequency of each CPU

2.2 Optimized GAN Training

2.2.1 Problem formulation

my proposed resource-constrained GAN optimization model, OGAN is as follow:

$$\min_{\mathbf{z},\mathbf{m},\mathbf{c}} \sum_{i=1}^{N} \tilde{\beta}_i((1-z_i)T_i^{max} + z_i(T_i^{com} + T_i^{gan} + T_i^{tr} + \alpha(\frac{\bar{V}_{N_\Upsilon}}{\sqrt{m_i}} + \frac{\bar{W}_{N_\Upsilon}}{\sqrt{N_{\Upsilon i}}}))) \tag{2.4a}$$

s.t.

$$z_i(T_i^{com} + \frac{m_i.\eta_i^{(g)}}{F_g} + \frac{\Phi_i.\eta_i^{(c)}}{F_c.c_i}) \leq T_i^{max}, \quad \forall i \in \mathcal{N}, \tag{2.4b}$$

$$z_i(\frac{\bar{V}_{N_\Upsilon}}{\sqrt{m_i}} + \frac{\bar{W}_{N_\Upsilon}}{\sqrt{N_{\Upsilon i}}}) \leq E_i^{min}, \quad \forall i \in \mathcal{N}, \tag{2.4c}$$

$$\sum_{i=1}^{N} z_i.R_i \leq R, \tag{2.4d}$$

$$\sum_{i=1}^{N} z_i.B_i \leq B, \tag{2.4e}$$

$$\sum_{i=1}^{N} c_i \leq N_c, \tag{2.4f}$$

$$0 \leq c_i \leq \Psi_i, \quad \forall i \in \mathcal{N}, \tag{2.4g}$$

$$\sum_{i=1}^{N} z_i \leq N_g, \tag{2.4h}$$

$$1 \leq m_i, \quad \forall i \in \mathcal{N}, \tag{2.4i}$$

$$z_i \in \{0,1\}, \quad \forall i \in \mathcal{N}. \tag{2.4j}$$

The objective function in (2.4a) minimizes two components. The first component, $(1-z_i)T_i^{max}$, represents the time DRL i operates without fine-tuning its parameters using a GAN. In the case of $z_i = 1$, the second component, $z_i(T_i^{com} + T_i^{gan} + T_i^{tr} + \alpha(\frac{\bar{V}_N}{\sqrt{m_i}} + \frac{\bar{W}_{N_\epsilon}}{\sqrt{N_{\epsilon i}}}))$, achieves a trade-off between two conflicting objectives: reducing E_i and the total training time of GAN i and DRL

Table 2.3 OGAN's decision variables

m_i	Number of training iterations used for GAN i
c_i	Number of CPU cores allocated for pre-training DRL i
z_i	1 if GAN i is trained, 0 otherwise

i, using the weight parameter α. It is necessary to allocate a greater number of resources to training GAN i when $\tilde{\beta}_i$ is higher, as it indicates greater unreliability in communication with SBS i. Notably, more resources ($\mathbf{m}$, $\mathbf{c}$) reduce the training time (GAN, DRL), resulting in greater URLLC reliability.

Constraint (2.4b) ensures the training time is within its allocated episode time. For a given E_i^{min}, constraint (2.4c) guarantees that GAN i undergoes a minimum number of training iterations, m_i, to prevent low-quality GANs. Bandwidth limit for uploading all of GAN training data from MBS to the cloud is indicated by constraint (2.4d). Constraints (2.4e) limit the total available shared wireless backhaul bandwidth from SBSs to MBS. Constraint (2.4f) limits the CPU usage of the MEC and each DRL. Constraint (2.4h) limits the number of GANs to the number of available GPUs and defines the minimum value of m_i. Since I assume a time-sharing policy for CPU cores, c_i is a positive real number.

2.2.2 Difference of Convex Programming

The optimization problem associated with OGAN is inherently challenging due to its nature as a mixed-integer non-convex constrained optimization problem. Directly solving such a problem is computationally demanding and often infeasible. To address this challenge, I propose adopting the Difference of Convex (DC) Functions programming approach, as outlined in (Lipp & Boyd (2016)). In this section, I begin by introducing DC programming and the algorithms utilized to solve such problems. Subsequently, I outline my proposed algorithm for solving OGAN in the following sections.

A function is called DC if we can express it as a difference of two convex functions. To elaborate, consider convex set Ω with functions $\mathcal{F}$, and convex functions $\mathcal{F}_1$ and $\mathcal{F}_2$ on it. If we have $\mathcal{F} = \mathcal{F}_1 - \mathcal{F}_2$, then $\mathcal{F}$ is a DC and $\mathcal{F}_1$ and $\mathcal{F}_2$ are called its convex components (An & Tao (2005)).

The general DC program does not inherently guarantee global solutions. However, empirical evidence indicates that when initialized from an appropriate starting point, it often converges to global solutions. The DC Algorithm (DCA), also known as the convex–concave procedure, has demonstrated success in solving various nonconvex optimization problems in practice. It is a powerful heuristic method used to find local solutions to DC programming problems (Yuille & Rangarajan (2003)). DCA is recognized for its ability to converge to a stationary point of the objective function under certain assumptions. This makes DCA a popular choice for optimizing complex nonconvex functions and finding solutions that meet practical requirements (Abbaszadehpeivasti, de Klerk & Zamani (2023)). Particularly in large-scale problems, it consistently produces global solutions that are more reliable and efficient compared to comparable standard methods.

DCA replaces the concave components of the problem ($\mathcal{F}_2$) with the linear part of their Taylor expansion. The Taylor expansion of a function is an infinite series representation of a function as a sum of its derivatives at a single point. For a function $f(x)$ that is infinitely differentiable at a point a, the Taylor series expansion around a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

2.2.3 DC Programming Solutions for OGAN

This method involves decomposing the objective function into the difference of two convex functions, thereby enabling the utilization of convex optimization techniques to efficiently solve the problem. By employing the DC programming approach, I aim to streamline the optimization process for OGAN and facilitate of finding the optimal solutions in a more computationally

tractable manner. Initially, I transform OGAN into a DC programming problem (DC-OGAN) by changing constraint (2.4j) to two new DC constraints:

$$z_i^2 - z_i \leq 0; \quad z_i - z_i^2 \leq 0, \quad i \in \mathcal{N}. \tag{2.5}$$

Based on general idea behind DC programming, to design a DC algorithm, I need convex components of the objective and constraint functions. Clearly, constraints (2.4d) (2.4e), (2.4f), (2.4g) and (2.4h) are convex, while constraint (2.4i) is concave. I identify the convex components of the other DC functions in the DC-OGAN problem by decomposition property described in Remark 1, as well as the theorem 1, stating that the sum of convex functions is itself convex.

Remark 1. *The product of two non-negative convex functions* $f_1, f_2 : \mathbb{R}^n \longrightarrow \mathbb{R}_+$*, can be decomposed to DC as*

$$f_1.f_2 = \frac{1}{2}[(f_1 + f_2)^2 - ((f_1)^2 + (f_2)^2)]$$

Theorem 1. *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *and* $g : \mathbb{R}^n \to \mathbb{R}$ *be two convex functions. Then, their sum* $h : \mathbb{R}^n \to \mathbb{R}$ *defined by*

$$h(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$$

is also a convex function.

Finally, using theorem 1 and remark 1, the OGAN problem is transformed into its DC form called DC-OGAN by incorporating these new constraints since the other functions are convex as stated above.

Algorithm 2.2 iteratively solves DC-OGAN using the Convex-Concave Procedure (CCP) (Lipp & Boyd (2016)). In each iteration, my algorithm transforms DC-OGAN into an equivalent convex form by approximating the second convex components of DC functions (consist of the objective function (Eq.(2.4a)), constraints (Eq.(2.4b) and Eq.(2.4c)) as well as two new functions in Eq.(2.5), added to replace the binary variable z) using the linear part of their Taylor expansions. This approximation replaces the concave components with linear approximations. Since the first part of the functions are convex and the second part are linear, the result is a

convex function that can be solved with any standard convex solver such as CVX. Subsequently, we iteratively update the optimal values of **m**, **z**, and **c** until reaching the stopping criteria, where the improvement in objective value is less than δ.

Algorithm 2.2 DC Algorithm (DCA)

Input: Given T_i^{max}, β_i, α, $\eta_i^{(g)}$, F_g, Φ_i, $\eta_i^{(c)}$, F_c, B_i, B, N_0, h_i, N_g, N_c, F_g, F_c, Ψ_i,
Initial feasible points ($k_{i,0}$, $m_{i,0}$, $c_{i,0}$; $\forall i \in \mathcal{N}$),
Stop criteria δ, $k := 0$

Output: m^*, c^*, z^*

25 **repeat**

1. Convexify:
   a. Forms the Taylor expansion of concave components (Eq.(2.4a), Eq.(2.4b), Eq.(2.4c), Eq.(2.5))
   b. Replace all the concave components with their Taylor expansion (linear approximation).
2. Solving the obtained convex problem using CVXPY.
3. Setting the solutions m as m_{k+1}, z as z_{k+1} and c as c_{k+1}.
4. Updating iteration; $k := k + 1$.

26 **until** *Stop criteria satisfied*;

Return m^*, c^*, z^*

CHAPTER 3

NUMERICAL RESULTS

In this chapter, I explore the results obtained from my research. Beginning with an overview of the simulation setup and tools used, I then detail my experiment before presenting and analyzing the results.

3.1 Simulation Settings

I simulate a UDN comprising N SBSs, each serving three users. The UDN provides URLLC service with a transmission rate of 400 packets per second per user, and a delay threshold of 3 milliseconds. Using Proximal Policy Optimization (PPO) in OpenAI Gym libraries (Brockman *et al.* (2016)), I implement DRLs and the simulation environment. GAN training occurs in the Google Colab environment using Nvidia Tesla A100 GPU with the PyTorch library. The number of required operation for training DRLs and GPUs are measured offline (bring more details).

Due to GPU resource limitations, I simulate the presence of N_g available GPUs (while I actually have only one) by employing time-sharing to train multiple GANs on a single GPU. Similarly, I use the same strategy for CPU cores to simulate having more CPU cores than I actually possess. Although my server has fewer CPU cores than the maximum available (N_c), I allocate the server's CPU cores using time-sharing to mimic the concurrent training of multiple DRL models. Specifically, my server has Ψ_i (64) CPU cores, which equals the maximum number of CPU cores that can be used for training a DRL model in a virtual environment.

In practice, virtualization technologies such as virtual machines or containers (e.g., Docker) can be employed to isolate resources. These technologies help in managing and distributing the computational workload, ensuring that concurrent computations minimally affect each other. Virtual machines provide a way to run multiple operating systems on a single physical machine, each with its allocated resources. Containers, on the other hand, offer a lightweight alternative by sharing the same operating system kernel while isolating the application processes. Both methods enhance the efficiency and flexibility of resource management, making it possible to simulate a

Table 3.1 Simulation parameters for experiment 1, 2 and 3

Variable	Value	Variable	Value	Variable	Value
T_i^{max}	2,000	B	400 MHz	B_i	40 MHz
E_i^{min}	0.1	N_g	6	$\eta_i^{(g)}$	52 KHz
F_c	3.5 GHz	$\eta_j^{(c)}$	480 KHz	N_c	256
F_g	1.4 GHz	Ψ_i	64	R	100 Mbps

high-resource environment even with limited physical hardware. Thus, the performance and execution time of my resource-sharing method for GPU and CPU are comparable to those achieved by running multiple virtual machines simultaneously, as both approaches effectively partition and manage available resources to optimize computational efficiency and throughput.

Based on my dataset and the specifics of my experiment, using Pytorch, I implemented my GAN (Generator and Discriminator) model as follows:

- The Generator architecture consists of 4 convolutional layers, each followed by a ReLU activation function. The final layer is a fully connected layer with a Tanh activation function, which helps in generating the output data in the desired range.
- The Discriminator architecture also consists of 4 convolutional layers, each followed by a ReLU activation function. The final layer is a fully connected layer, which is followed by a Sigmoid activation function to produce the probability output, indicating whether the input data is real or generated.

I carry out four distinct experiments to assess OGAN:

- In the first experiment, I conducted 10 trials using various channel gain distributions (between users and their corresponding SBS) and constant user traffic to test the performance of DRL and the reliability of URLLC. To compare the performance of OGAN with respect to baseline methods, I present the cumulative rewards of all DRLs and the cumulative number of packet losses as a metric for reliability, averaged across all 10 trials. Each trial has a duration of 1500 seconds, with other simulation parameters detailed in Table 3.1.

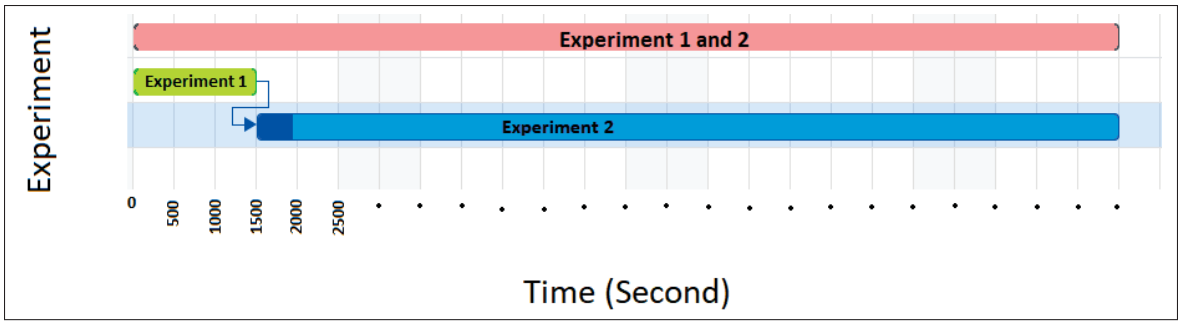

Figure 3.1 Gantt chart illustrating the relationship between experiments 1 and 2

- In a subsequent experiment, I examine the generalization capabilities of DRLs in adapting to environmental changes. As depicted in Figure 3.1, following the completion of the initial experiment, I extend the trials using the same setup until the 2,000th second. At this point, I abruptly alter the environment to gauge its impact on DRLs. This involves reducing the channel gain for all users, creating an extreme environment for the DRLs. Subsequently, in order to enhance the performance of the DRLs and adapt them to the new environment, I retrain the models with a new dataset for fine-tuning. My objective is to evaluate whether my proposed approach improves the recovery speed of DRLs in response to sudden environmental changes, with the results presented in Section 3.2.
- To demonstrate the impact of resource limitations on the overall reliability of URLLC, I explore the consequences of reducing the value of T_i^{max} on URLLC reliability. Due to the decrease in T_i^{max}, a tight deadline is imposed on both data transmission and GAN/DRL training, necessitating additional computational resources for each round of training. T_i^{max} is generated from a normal distribution, $T_i^{max} \sim \mathcal{N}(\mu, \sigma^2)$, where $\sigma^2 := \mu/2$. By decreasing the value of μ, I perform 900-second trials with other parameters listed in Table 3.1. By comparing the total packet loss of OGAN with that of other baselines, I determine the performance of OGAN.
- In the fourth experiment, I consider the transmission time, the duration required to complete the training of the DRL model (after its GAN has been trained) for an SBS. To achieve this, I measure the average transmission time across all SBSs. In previous experiments, it was possible to train the GAN and DRL models for some SBSs multiple times, while others might

Table 3.2 Simulation parameters for experiment 4

Variable	Value	Variable	Value	Variable	Value
T_i^{max}	12,000	B	N × 40 MHz	B_i	40 MHz
E_i^{min}	0.1	N_g	10	$\eta_i^{(g)}$	52 KHz
F_c	3.5 GHz	$\eta_j^{(c)}$	480 KHz	N_c	256
F_g	1.4 GHz	Ψ_i	64	R	10× N Mbps
N	(10, 25, 50, 75, 100)	Experiment duration	(1500, 2500, 3500, 4500, 6000)		

not have been trained at all. In this experiment, I assume that for each SBS, I train GAN and DRL maximum one time. This experiment demonstrates the performance of OGAN as the scale of the problem increases by adding more SBSs (N), selected from the list (10, 30, 50, 70, 90). The bandwidth from the MBS to the cloud is calculated as $N \times 10$ MHz, and the shared wireless backhaul is $N \times 40$ MHz. While the above parameters increase with the number of SBSs, other resources such as the number of GPUs (N_g) and CPU cores (N_c) remain constant at values of 10 and 256, respectively. Other simulation parameters are listed in Table 3.2.

The duration of experiment corresponding to each parameter of the SBSs list are (1500, 2500, 3500, 4500, 6000) respectively. Even after finishing the training of GAN and DRLs, I continoue the experiment 4 until above times to measure and compare the reliability for each of them. During the experiment, by increasing the number of SBSs from the above list, I compare the average transmission time between OGAN and two other baseline methods. This comparison allows us to evaluate the scalability and efficiency of OGAN in handling an increasing number of SBSs.

In the first three experiments, the distribution of the anomalous dataset kept changing. This variability meant that once a GAN was trained for a specific SBS, the rare samples it generated did not match the anomalous samples encountered later. Consequently, it is always required to train new GANs for each newly generated anomalous dataset. However, in the fourth experiment, I assume the distribution of the anomalous dataset remained constant so that

once I train a GAN for a SBS, I don't retrain another GAN/DRL after that and SBSs that have trained GAN/DRL are removed from pool.

Note that, since the number of SBSs exceeds the number of available GPUs, it is impossible to train all SBSs in one round. The maximum number of SBSs selected for training each time OGAN is called is equal to the number of GPUs, N_g. Whenever the training of a GAN model on a GPU is completed, or when the training of one of the DRL models finishes and CPU cores are released, OGAN is called again to determine resource allocation and assign resources for training the current GANs or choosing a new one.

3.1.1 Dataset

My assumption for every DRL is that users' channel gain will remain constant during the duration of that episode. The OFDMA matrices for the DRL state-space are generated based on a Rayleigh fading distribution and are referred to as normal datasets. To simulate the effect of rare samples, I use abnormal datasets created by adding random deep fades to the channel gain of some users during the simulation process. The proportion of abnormal to normal samples for SBSs is randomly selected from the list $\{0.01, 0.05, 0.1\}$. Each DRL is trained using its normal dataset before being put through the simulation phase. During the experiment, for each SBS, 10^5 samples of the normal dataset are used for the state-space of DRLs, while abnormal samples are shuffled into the state-space.

3.1.2 Baselines

I compare OGAN's performance with two baseline methods: the modified time fairness scheme (Fair) and Continuous Edge Learning (CEL), presented as follows.

The first baseline is a modified time fairness scheme (Fair) (Zhou *et al.* (2020)), which allocates bandwidth, GPUs, and CPU cores equally across all SBSs. Over time, all SBSs eventually receive equal GPU and CPU resources, starting with the less reliable ones to first train GAN/DRL

models for them and increase their reliability, thereby enhancing overall system reliability. GAN training continues until convergence or reaches 2×10^6 training iterations.

The second baseline is adapted from the Continuous Edge Learning (CEL) model (Xie *et al.* (2023)), originally intended for continuous DNN training in resource-limited edge environments. By excluding monetary computation costs from the objective function, employing deterministic input parameters, and considering limited cloud resources, CEL aligns with OGAN's aim to enhance the performance of DRLs and GANs.

3.2 Numerical Results

As the primary objective of OGAN is to enhance reliability by reducing packet loss, I evaluate the reliability by adding up cumulative packet loss across all SBSs. Figure 3.2 shows the cumulative packet loss data for 1500 seconds of system operation. Since the DRLs are pre-trained before the first experiment, the initial performance across all three baselines remains consistent, given that the same DRL is used for each SBS across approaches. Nonetheless, the distinctions become more evident over time according to the varying order, frequency, and GANs employed during DRL training. The pivotal point of differentiation occurs around second 400, marking the completion of the first round of GAN/DRL training and the replacement of old, adapted DRLs with new, adapted one. My findings show that OGAN led to a 23% increase in overall system reliability, surpassing the second-best baseline, CEL. The reliability improvement is attributed to optimized resource allocation and enhanced quality of trained GANs. OGANs minimum training dataset requirement, e.g., 1,000 samples in my experiment, reduces the likelihood of GAN training issues, such as divergence and mode collapse. This increases the dataset diversity and, subsequently, improves the performance and reliability of DRLs.

In Figure 3.3, OGAN results in a 22% increase in the sum of rewards for all DRLs compared to CEL, and a 45% increase compared to Fair during my first experiments. The baselines exhibit relatively small differences in performance when the initial rounds of GAN and DRL training are still in progress. However, after multiple training rounds, the reward disparities

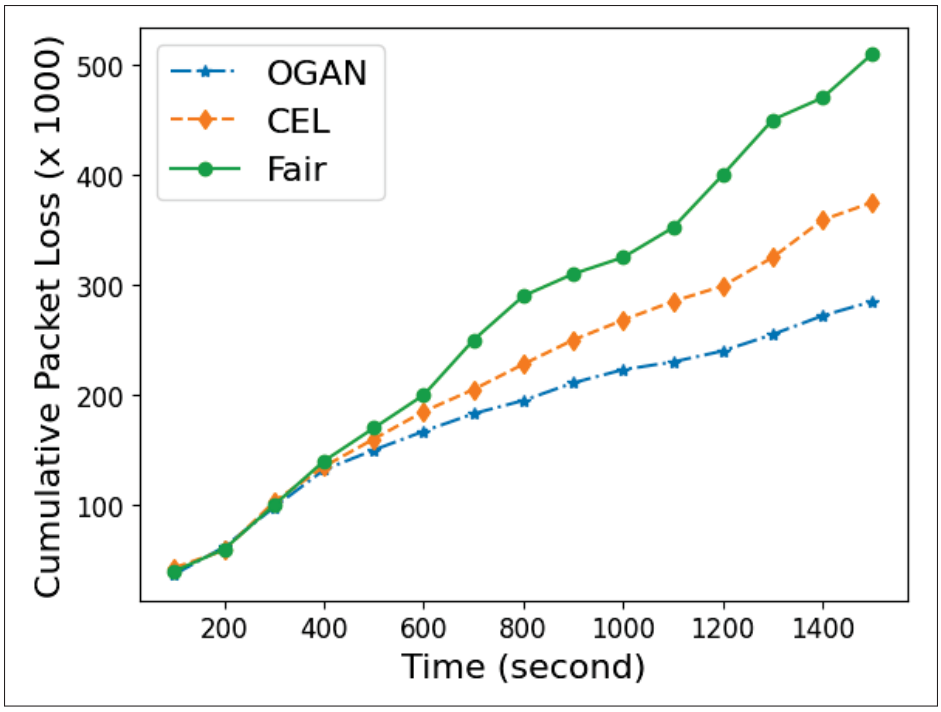

Figure 3.2 Cumulative Packet Loss Comparison: The differences between the schemes become noticeable around the 400-second mark, where OGAN completes the first batch of DRL training with datasets generated by GANs, leading to a significant reduction in packet loss compared to CEL and Fair

become increasingly apparent. Both optimal resource allocation and the training of GANs are responsible for the superior results of OGAN. As the reward in my DRL model is impacted by the sum of transmission energy and packet loss, the superior performance of OGAN can demonstrate an increase in reliability and a reduction in energy consumption.

Fig. 3.4 illustrates the impact of varying the mean value (μ) of T_i^{max}, representing the wireless episode time, ranging from 10^3 to 10^2. As observed, the total packet loss decreases across all schemes, with OGAN exhibiting the smallest loss. The graph indicates that OGAN maintains higher reliability compared to CEL, showing an 18% decrease in reliability for lower T_i^{max} values, while this reduction is approximately 12% for higher T_i^{max} values. The superior performance of OGAN can be attributed to its resource allocation mechanism, which strategically focuses on training a smaller subset of SBSs in resource-constrained environments, as opposed to distributing

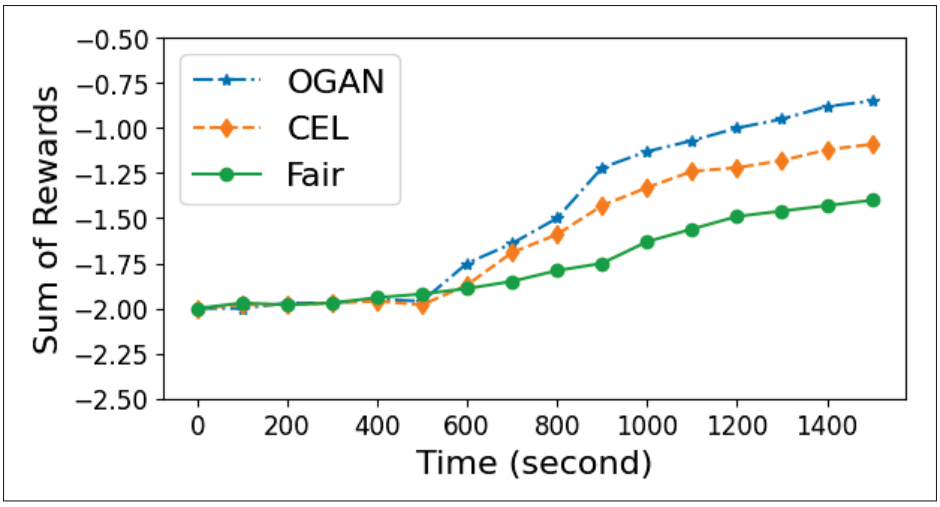

Figure 3.3 Sum of rewards for all DRLs. DRLs attempt to increase the negative rewards. Trained agents are deployed at time zero to initiate the experiment. Performance improvement begins around the 500-second mark, resulting in gradual growth in rewards

Table 3.3 Number of trained GANs/DRLs before 2000th second

Method	**OGAN**	**CEL**	**Fair**
Number of Rounds	24	26	19

resources among the maximum possible SBSs. By incorporating T_i^{max} into my optimization model, OGAN accounts for sudden predicted changes in the environment. Consequently, OGAN avoids allocating resources to SBSs with small T_i^{max} where their episodes are nearing completion. Instead, it prioritizes the training with higher values of T_i^{max}. In contrast, CEL and Fair do not dynamically adapt to sudden task changes and persist in training GANs and DRLs using outdated datasets, irrespective of T_i^{max}.

Fig. 3.5 shows the result from the generalization power of DRLs when using different resource allocation schemes to train GAN for them. As shown in table 3.3, prior to the second 2,000, OGAN had 24 rounds of trained GANs and DRLs, CEL had 26, and Fair had 19 rounds. As seen in second 2,000 (equal to iteration 0 in Figure 3.5), all three approaches converge to

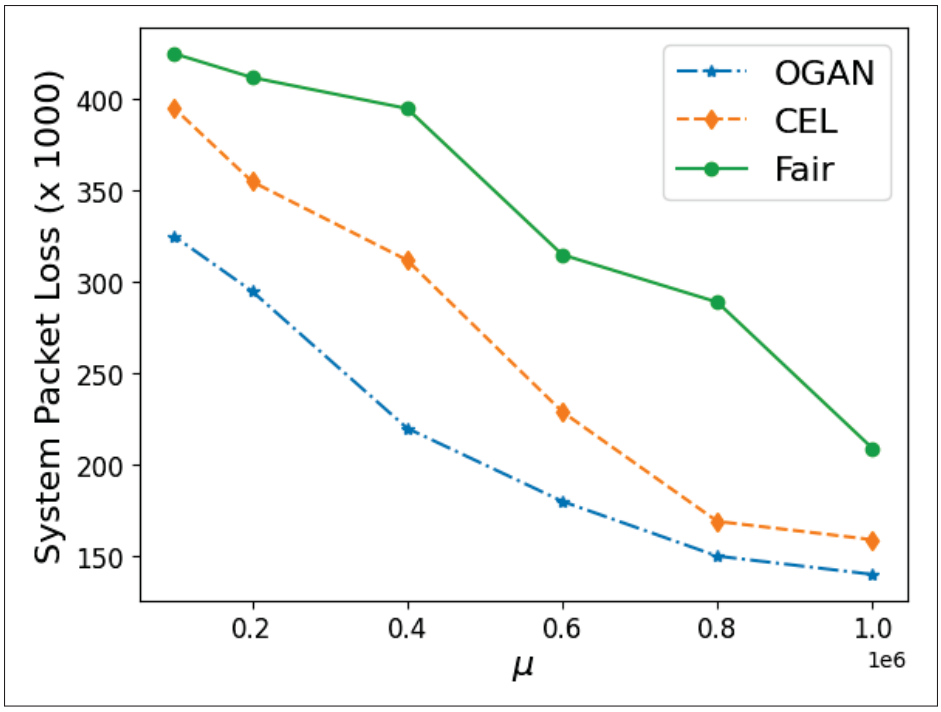

Figure 3.4 Impact of varying the mean value (μ) of T_i^{max} on system packet loss. The graph shows that as μ increases, the total packet loss decreases across all schemes, with OGAN exhibiting the smallest loss.

their current environment just prior to environment change. However, a sudden decrease in the channel gain at iteration 4,000 results in sharp decrease of rewards of all DRLs with the minimum performance reduction attributed to OGAN. Then, all agents are trained based on the dataset from the new environment to recover them by fine-tuning their DNNs. While the recovery time for DRLs trained based on OGAN takes around 7,000 iterations, for CEL, this value is 10,000, while Fair takes more than 13,000 iterations to converge. This improvement is for more generalization power DRLs gained during OGAN. Training more high-quality GANs results in a more diverse dataset for DRLs, reducing overfitting and enhancing the generalization. Consequently, higher episode returns are gained, and the adaptation time to converge to the new environments is decreased. This result is critical for URLLC since it shows a decrease in the period of unreliability and the recovery time after sudden changes in DRL state-space (Kasgari *et al.* (2020)).

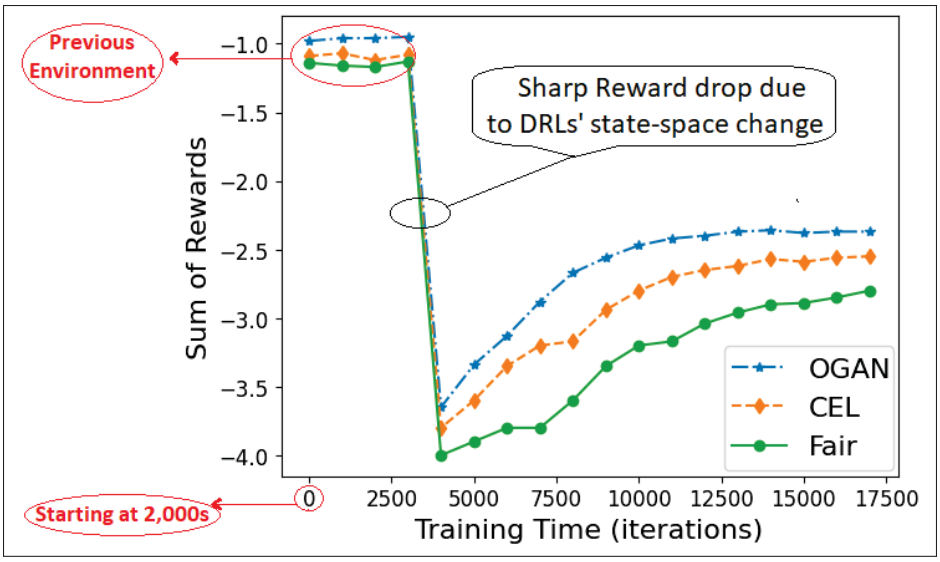

Figure 3.5 DRL Resilience to Environment Change: Iteration zero continues from the first experiment, which ended after 2,000 seconds. A sudden channel gain decrease at iteration 4,000 causes a sharp reward drop. Recovery times and reward drop in OGAN is less than other baselines.

Figure 3.6 shows the average transmission time from the fourth experiment. I increase the scale of the problem by increasing the number of SBSs (from list (10, 30, 50, 70, 90)) and then measure the average transmission time of all SBSs in OGAN and two baselines. As the number of SBSs increases, the computation resources (GPU and CPU cores) remain constant, causing OGAN to be called multiple times to train all GANs and DRLs in multiple rounds. Specifically, Figure 5 illustrates that with 10 SBSs, the average transmission times for OGAN, CEL, and Fair are similar, with OGAN being slightly lower. However, the gap between OGAN and the other baselines increases as the number of SBSs increases. This demonstrates that as the problem scale increases, OGAN maintains a more efficient average transmission time compared to the baseline methods.

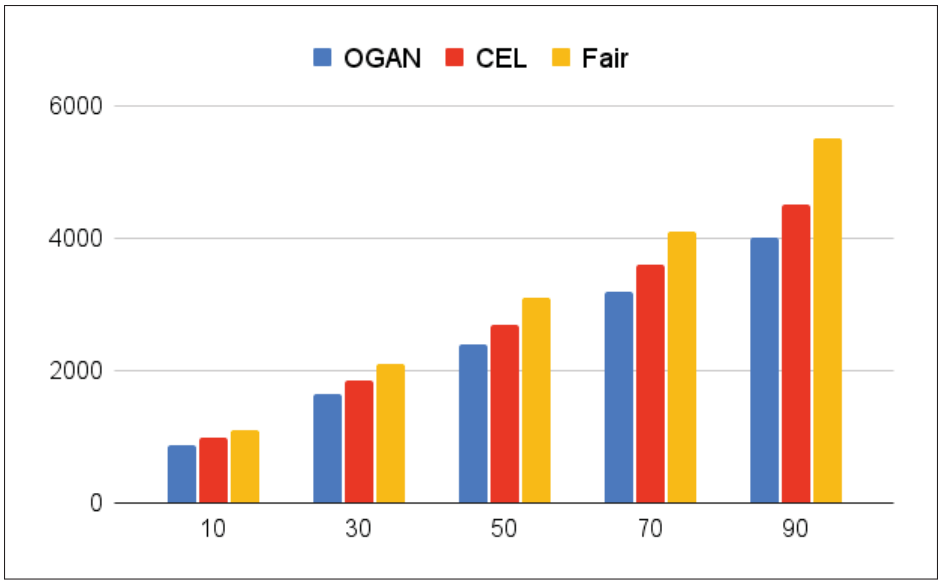

Figure 3.6 Average transmission time for OGAN, CEL, and Fair as the number of SBSs increases. OGAN consistently outperforms the baseline methods, particularly at larger scales.

CONCLUSION AND RECOMMENDATIONS

In this thesis, I addressed the problem of resource allocation and sharing for training GANs and DRLs in resource-limited ultra-dense 5G networks. I presented an architecture for resource sharing in ultra-dense 5G networks that facilitates the concurrent training of multiple GANs on the cloud and edge. In my architecture, GANs are trained in cloud GPUs, and DRLs are fine-tuned by training them in a simulator at the edge to improve the reliability of URLLC services when using DRLs for OFDMA resource allocation.

To enhance DRL reliability, I introduced an optimization model called OGAN, which considers computational and communication constraints in ultra-dense 5G networks, at both the edge and the cloud. Since OGAN is a mixed-integer non-convex problem, I developed an algorithm based on DC programming to find its solution in a timely manner that fits the constraints of 5G networks. Through extensive simulations, I demonstrated a substantial improvement in URLLC reliability using OGAN compared to two other baseline methods.

Further research could incorporate additional factors, such as memory constraints in the cloud and edge for storing training datasets, distributing the training of GANs across multiple GPUs, parallel training of DRLs on multiple edge servers, and deciding whether to perform the training of DRLs on the cloud or MEC.

BIBLIOGRAPHY

Abbaszadehpeivasti, H., de Klerk, E. & Zamani, M. (2023). On the rate of convergence of the difference-of-convex algorithm (DCA). *Journal of Optimization Theory and Applications*, 194(3), 605–626.

Ali, R., Zikria, Y. B., Bashir, A. K., Garg, S. & Kim, H. S. (2021). URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence. *IEEE Access*, 9, 67064–67095.

Alsenwi, M., Tran, N. H., Bennis, M., Pandey, S. R., Bairagi, A. K. & Hong, C. S. (2021). Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach. *IEEE Transactions on Wireless Communications*, 20(7), 4585–4600.

Alvarado, A., Scutari, G. & Pang, J.-S. (2014). A new decomposition method for multiuser DC-programming and its applications. *IEEE Transactions on Signal Processing*, 62(11), 2984–2998.

An, L. T. H. & Tao, P. D. (2005). The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133, 23–46.

Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein generative adversarial networks. *International conference on machine learning*, pp. 214–223.

Arndt, K., Hazara, M., Ghadirzadeh, A. & Kyrki, V. (2020). Meta reinforcement learning for sim-to-real domain adaptation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2725–2731.

Ayanoglu, E., Davaslioglu, K. & Sagduyu, Y. E. (2022). Machine learning in NextG networks via generative adversarial networks. *IEEE Transactions on Cognitive Communications and Networking*, 8(2), 480–501.

Bang, D. & Shim, H. (2021). Mggan: Solving mode collapse using manifold-guided training. *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2347–2356.

Berkenkamp, F., Turchetta, M., Schoellig, A. & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Casasole, B., Bonati, L., D'Oro, S., Basagni, S., Capone, A. & Melodia, T. (2021). QCell: Self-optimization of Softwarized 5G Networks through Deep Q-learning. *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 01–06.

Chan, S. C., Lampinen, A. K., Richemond, P. H. & Hill, F. (2022). Zipfian environments for reinforcement learning. *Conference on Lifelong Learning Agents*, pp. 406–429.

Davaslioglu, K. & Sagduyu, Y. E. (2018). Generative adversarial learning for spectrum sensing. *2018 IEEE international conference on communications (ICC)*, pp. 1–6.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S. & Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9), 2419–2468.

Eghbali, Y., Taskou, S. K., Mili, M. R., Rasti, M. & Hossain, E. (2024). Providing URLLC Service in Multi-STAR-RIS Assisted and Full-Duplex Cellular Wireless Systems: A Meta-Learning Approach. *IEEE Communications Letters*, 28(7), 1526–1530.

Ge, X., Tu, S., Mao, G., Wang, C.-X. & Han, T. (2016). 5G ultra-dense cellular networks. *IEEE Wireless Communications*, 23(1), 72–79.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gu, R. & Zhang, J. (2019). GANSlicing: A GAN-based software defined mobile network slicing scheme for IoT applications. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7.

Gui, J., Sun, Z., Wen, Y., Tao, D. & Ye, J. (2021). A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 35(4), 3313–3332.

Gujarati, A., Karimi, R., Alzayat, S., Hao, W., Kaufmann, A., Vigfusson, Y. & Mace, J. (2020). Serving DNNs like clockwork: Performance predictability from the bottom up. *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pp. 443–462.

Hardy, C., Le Merrer, E. & Sericola, B. (2019). Md-gan: Multi-discriminator generative adversarial networks for distributed datasets. *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, pp. 866–877.

Hellaoui, H., Yang, B. & Taleb, T. (2021). Towards using Deep Reinforcement Learning for Connection Steering in Cellular UAVs. *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 01–06.

Ho, T. M., Nguyen, T. T., Nguyen, K.-K. & Cheriet, M. (2021). Deep reinforcement learning for URLLC in 5g mission-critical cloud robotic application. *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6.

Hua, Y., Li, R., Zhao, Z., Chen, X. & Zhang, H. (2019). GAN-powered deep distributional reinforcement learning for resource management in network slicing. *IEEE Journal on Selected Areas in Communications*, 38(2), 334–349.

Ji, K., Zhou, Y. & Liang, Y. (2021). Understanding estimation and generalization error of generative adversarial networks. *IEEE Transactions on Information Theory*, 67(5), 3114–3129.

Jia, L., Zhou, Z., Xu, F. & Jin, H. (2021). Cost-efficient continuous edge learning for artificial intelligence of things. *IEEE Internet of Things Journal*, 9(10), 7325–7337.

Kasgari, A. T. Z. & Saad, W. (2019). Model-free ultra reliable low latency communication (URLLC): A deep reinforcement learning framework. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6.

Kasgari, A. T. Z., Saad, W., Mozaffari, M. & Poor, H. V. (2020). Experienced deep reinforcement learning with generative adversarial networks (GANs) for model-free ultra reliable low latency communication. *IEEE Transactions on Communications*, 69(2), 884–899.

Khan, B. S., Jangsher, S., Ahmed, A. & Al-Dweik, A. (2022). URLLC and eMBB in 5G industrial IoT: A survey. *IEEE Open Journal of the Communications Society*, 3, 1134–1163.

Khan, S. H., Hayat, M. & Barnes, N. (2018). Adversarial training of variational auto-encoders for high fidelity image generation. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1312–1320.

Kiumarsi, B., Vamvoudakis, K. G., Modares, H. & Lewis, F. L. (2017). Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, 29(6), 2042–2062.

Lei, L., Yuan, Y., Vu, T. X., Chatzinotas, S., Minardi, M. & Montoya, J. F. M. (2021). Dynamic-adaptive AI solutions for network slicing management in satellite-integrated B5G systems. *IEEE Network*, 35(6), 91–97.

Li, X., Li, D., Wan, J., Vasilakos, A. V., Lai, C.-F. & Wang, S. (2017). A review of industrial wireless networks in the context of industry 4.0. *Wireless Networks*, 23(1), 23–41.

Lipp, T. & Boyd, S. (2016). Variations and extension of the convex–concave procedure. *Optimization and Engineering, 17 (2)*.

Meng, F., Chen, P., Wu, L. & Cheng, J. (2020). Power allocation in multi-user cellular networks: Deep reinforcement learning approaches. *IEEE Transactions on Wireless Communications*, 19(10), 6255–6267.

Mismar, F. B., Evans, B. L. & Alkhateeb, A. (2019). Deep reinforcement learning for 5G networks: Joint beamforming, power control, and interference coordination. *IEEE Transactions on Communications*, 68(3), 1581–1592.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540), 529–533.

Navarro-Ortiz, J., Romero-Diaz, P., Sendra, S., Ameigeiras, P., Ramos-Munoz, J. J. & Lopez-Soler, J. M. (2020). A survey on 5G usage scenarios and traffic models. *IEEE Communications Surveys & Tutorials*, 22(2), 905–929.

Navidan, H., Moshiri, P. F., Nabati, M., Shahbazian, R., Ghorashi, S. A., Shah-Mansouri, V. & Windridge, D. (2021). Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation. *Computer Networks*, 194, 108149.

Nguyen, C. T., Van Huynh, N., Chu, N. H., Saputra, Y. M., Hoang, D. T., Nguyen, D. N., Pham, Q.-V., Niyato, D., Dutkiewicz, E. & Hwang, W.-J. (2021). Transfer learning for future wireless networks: A comprehensive survey. *arXiv preprint arXiv:2102.07572*.

Ong, K. S. H., Wang, W., Niyato, D. & Friedrichs, T. (2021). Deep-reinforcement-learning-based predictive maintenance model for effective resource management in industrial IoT. *IEEE Internet of Things Journal*, 9(7), 5173–5188.

O'Shea, T. J., Roy, T., West, N. & Hilburn, B. C. (2018). Physical layer communications system design over-the-air using adversarial networks. *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 529–532.

Polese, M., Jana, R., Kounev, V., Zhang, K., Deb, S. & Zorzi, M. (2020). Machine learning at the edge: A data-driven architecture with applications to 5G cellular networks. *IEEE Transactions on Mobile Computing*, 20(12), 3367–3382.

Radford, A., Metz, L. & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rao, S. S. (2019). *Engineering optimization: theory and practice*. John Wiley & Sons.

Schulman, J. & et al. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shani, L., Efroni, Y. & Mannor, S. (2020). Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04), 5668–5675.

She, C., Dong, R., Gu, Z., Hou, Z., Li, Y., Hardjawana, W., Yang, C., Song, L. & Vucetic, B. (2020). Deep learning for ultra-reliable and low-latency communications in 6G networks. *IEEE network*, 34(5), 219–225.

Shen, Y., Shi, Y., Zhang, J. & Letaief, K. B. (2019). LORM: Learning to optimize for resource management in wireless networks with few training samples. *IEEE Transactions on Wireless Communications*, 19(1), 665–679.

Sun, H. & et al. (2022). Learning to Continuously Optimize Wireless Resource in a Dynamic Environment: A Bilevel Optimization Perspective. *IEEE Trans. on Signal Processing*.

Sun, H., Pu, W., Zhu, M., Fu, X., Chang, T.-H. & Hong, M. (2021). Learning to continuously optimize wireless resource in episodically dynamic environment. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949.

Tran, N. H., Bao, W., Zomaya, A., Nguyen, M. N. & Hong, C. S. (2019). Federated learning over wireless networks: Optimization model design and analysis. *IEEE INFOCOM 2019-IEEE conference on computer communications*, pp. 1387–1395.

Uesato, J. & et al. Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures. *ICLR 2019,*.

Wang, C.-X., Haider, F., Gao, X., You, X.-H., Yang, Y., Yuan, D., Aggoune, H. M., Haas, H., Fletcher, S. & Hepsaydir, E. (2014). Cellular architecture and key technologies for 5G wireless communication networks. *IEEE communications magazine*, 52(2), 122–130.

Wang, S., Wang, R., Hao, Q., Wu, Y.-C. & Poor, H. V. (2020). Learning centric power allocation for edge intelligence. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6.

Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B. & Miao, Q. (2022a). Deep reinforcement learning: a survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Wang, Z., Hu, J., Min, G., Zhao, Z., Chang, Z. & Wang, Z. (2022b). Spatial-temporal cellular traffic prediction for 5G and beyond: A graph neural networks-based approach. *IEEE Transactions on Industrial Informatics*, 19(4), 5722–5731.

Weng, H., Li, L., Cheng, Q., Chen, W. & Han, Z. (2020). Content caching policy based on GAN and distributional reinforcement learning. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–5.

Xiang, B., Elias, J., Martignon, F. & Di Nitto, E. (2019). Joint network slicing and mobile edge computing in 5G networks. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–7.

Xiao, Y., Shi, G., Li, Y., Saad, W. & Poor, H. V. (2020). Toward self-learning edge intelligence in 6G. *IEEE Communications Magazine*, 58(12), 34–40.

Xie, H., Xia, M., Wu, P., Wang, S. & Poor, H. V. (2023). Edge learning for large-scale Internet of Things with task-oriented efficient communication. *IEEE Transactions on Wireless Communications*, 22(12), 9517–9532.

Xiong, Z., Zhang, Y., Niyato, D., Deng, R., Wang, P. & Wang, L.-C. (2019). Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges. *IEEE Vehicular Technology Magazine*, 14(2), 44–52.

Xu, X., Liu, X., Yin, X., Wang, S., Qi, Q. & Qi, L. (2020). Privacy-aware offloading for training tasks of generative adversarial network in edge computing. *Information Sciences*, 532, 1–15.

Yang, Y., Li, Y., Zhang, W., Qin, F., Zhu, P. & Wang, C.-X. (2019). Generative-adversarial-network-based wireless channel modeling: Challenges and opportunities. *IEEE Communications Magazine*, 57(3), 22–27.

Ye, H., Liang, L., Li, G. Y., Kim, J., Lu, L. & Wu, M. (2018). Machine learning for vehicular networks: Recent advances and application examples. *ieee vehicular technology magazine*, 13(2), 94–101.

Yuan, Y., Jiao, L., Zhu, K., Lin, X. & Zhang, L. (2022). AI in 5G: The Case of Online Distributed Transfer Learning over Edge Networks. *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pp. 810–819.

Yuille, A. L. & Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, 15(4), 915–936.

Zhang, C., Dang, S., Shihada, B. & Alouini, M.-S. (2021a). Dual attention-based federated learning for wireless traffic prediction. *IEEE INFOCOM 2021-IEEE conference on computer communications*, pp. 1–10.

Zhang, J., Zhao, L., Yu, K., Min, G., Al-Dubai, A. Y. & Zomaya, A. Y. (2023). A novel federated learning scheme for generative adversarial networks. *IEEE Transactions on Mobile Computing*, 22(4), 1983–1997.

Zhang, Q., Ferdowsi, A., Saad, W. & Bennis, M. (2021b). Distributed conditional generative adversarial networks (GANs) for data-driven millimeter wave communications in UAV networks. *IEEE Transactions on Wireless Communications*, 21(3), 1438–1452.

Zhang, T., Zhu, K. & Niyato, D. (2019). A generative adversarial learning-based approach for cell outage detection in self-organizing cellular networks. *IEEE Wireless Communications Letters*, 9(2), 171–174.

Zhong, C., Gursoy, M. C. & Velipasalar, S. (2020). Deep reinforcement learning-based edge caching in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 6(1), 48–61.

Zhou, H., Erol-Kantarci, M. & Poor, H. V. (2022). Learning from peers: Deep transfer reinforcement learning for joint radio and cache resource allocation in 5G RAN slicing. *IEEE Transactions on Cognitive Communications and Networking*, 8(4), 1925–1941.

Zhou, L., Hong, Y., Wang, S., Han, R., Li, D., Wang, R. & Hao, Q. (2020). Learning centric wireless resource allocation for edge computing: Algorithm and experiment. *IEEE Transactions on Vehicular Technology*, 70(1), 1035–1040.

Zhu, Y. & Wang, S. (2021). Joint traffic prediction and base station sleeping for energy saving in cellular networks. *ICC 2021-IEEE International Conference on Communications*, pp. 1–6.