

HELLENIC MEDITERRANEAN UNIVERSITY

5G Technology for Public Safety; Enhancing Disaster Response with Advanced Communication Systems

by

Andreas Georgitsis - TH20071

A thesis submitted in partial fulfillment for the
Undergraduate degree

in the
Polytechnic School
Department of Electrical and Computer Engineering

Supervising Professor: Dr. Evangelos K. Markakis

Date

Declaration of Authorship

I, Andreas Georgitsis, declare that this thesis titled, 5G Technology for Public Safety; Enhancing Disaster Response with Advanced Communication Systems and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Andreas Georgitsis

Date: 30/09/2025

᾽Αρχὴ ἥμισυ παντός.

The beginning is half of the whole.

Plato

HELLENIC MEDITERRANEAN UNIVERSITY

Abstract

Polytechnic School
Department of Electrical and Computer Engineering

Undergraduate Degree

by Andreas Georgitsis - TH20071

This thesis proposes a semantic-aware Deep Reinforcement Learning framework for dynamic slicing in 5G networks. The system was implemented in NS-3.44 with the official NR module and the NS3-AI interface. This solution enables real-time interaction between a Deep Q-Network agent and the simulated network. The environment of the simulation includes three slices, one URLLC, one eMBB and one mMTC, mapped to distinct traffic types for each service. The agent receives from the environment through shared memory a 21 state vector that includes both physical KPIs and semantic indicators. Using the same shared memory infrastructure every 100ms, the agent sends allocation decisions to optimize both network performance and service intent alignment. Simulation results show URLLC latency below 2 ms, significant eMBB throughput and semantic scores above 0.95, outperforming baselines. The findings demonstrate that integrating semantic metrics into DRL-based slicing enhances adaptability, efficiency and QoS compliance in next-generation networks.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my family whose unwavering support and encouragement have been the foundation of my academic journey. My parents, Nikolaos and Anna and my sister, Stamatina have been a constant source of motivation and strength, one that enabled me to pursue my goals with determination even during stressful and difficult times.

I would also like to extend my sincere thanks to my supervisors: Professor Evangelos K. Markakis, Dr. Ilias Politis, and Dr. Tafseer Akhtar for their invaluable guidance and expertise that aided towards the completion of this thesis. Their insightful feedback, constructive advice, patience and focus on innovation have been instrumental in shaping the quality and direction of my work.

Finally, I am grateful to my friends, who have provided encouragement and were always compassionate and understanding during this journey. Their companionship has been a source of balance and joy throughout these last six years.

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 5G Service Landscape . . . 2
- 1.2 Network Slicing and Dynamic Resource Allocation . . . 3
- 1.3 Intent-Based Networking and Machine Learning . . . 4
- 1.4 Semantic Communication in 5G . . . 6

2 Literature Review 7

3 Notation &Fundamentals 17
- 3.1 Notation Table . . . 17
- 3.2 Deep Reinforcement Learning and DQN . . . 19
- 3.3 Intent-Based Networking (IBN) Integration . . . 24
- 3.4 Semantic Communication Integration . . . 26

4 Basic Principles and Implementation Framework for an intelligent DRL-based agent to support IBN capable network slicing with a semantic layer. 29
- 4.1 Problem Statement . . . 29
- 4.2 Proposed Solution Architecture . . . 30
- 4.3 System Workflow and Implementation . . . 37

5 Implementation and Core Components of the solution 39
- 5.1 Agent Interface, State &Action Definition, and Reward Shaping (5.1) . . . 39
  - Control interval. . . . 39
  - State vector. . . . 39
  - Action space and scheduler knobs. . . . 41

Intent targets and safety guardrails. 41
Reward shaping. 41
IBN/semantics coupling. 43
Summary. 43
5.2 Neural Network Architecture (5.2) 44
5.3 DQN Training Methodology in NS-3 45
5.4 Agent Interaction with the NS-3 Slicing Simulation 47

6 Experimentation &Validation **50**
6.1 Experiment Setup 50
6.2 Data Sources and Measurement Methodology 52
6.3 Performance Metrics &Analysis 54
Overview 54
Throughput Analysis 55
Latency Behaviour 56
Allocation Patterns 57
Learning Dynamics and Semantic Performance 58
Correlation of KPIs and Semantic Metrics 59
Aggregate Evaluation 60
6.4 Discussion of Observed Trends 61
6.5 Limitations 62
6.6 Summary of Experimental Findings 62

7 Conclusions &Future Work **63**
7.1 Summary of Work 63
7.2 Key Contributions 64
7.3 Future Work 65
7.4 Final Remarks 66

Bibliography **67**

List of Figures

1.1 High-level view of Intent-Based Networking (IBN) architecture and its integration with network slicing and automation. 5

2.1 Three-level model of semantic communication. 15

3.1 Co-simulation flow of the system. 24
3.2 Closed-loop IBN-DQN-Semantic Communication architecture. 25

4.1 Official representation of the Shared Memory interface of NS3-ai. 37
4.2 Complete end-to-end workflow for the system. 38

5.1 Block diagram of the attention-augmented, dueling DQN. 44

6.1 Initial per-slice throughput. 51
6.2 Distribution of semantic-layer metrics. 55
6.3 Boxplot of key KPIs across slices. 56
6.4 Per-slice latency over time. 57
6.5 Resource-allocation decision evolution. 58
6.6 Reward history 59
6.7 Semantic vs. baseline 59
6.8 Learning dynamics and semantic performance. 59
6.9 Correlation matrix between network KPIs and semantic-layer indicators. . . . 60
6.10 Aggregate distributions of key semantic KPIs and URLLC latency. 61

List of Tables

3.1 Notation Table 17
3.2 Semantic Indicators and Their Functional Roles 27

4.1 Comparison Between Conventional Approaches and the Proposed Solution . . 33

5.1 State vector features exported in `SlicingState` and consumed by the agent (order is exact). 40

Abbreviations

Acronym	Stands For
3GPP	Third Generation Partnership Project
AI	Artificial Intelligence
ARP	Allocation and Retention Priority
BWP	Bandwidth Part
CBR	Constant Bit Rate
CNN	Convolutional Neural Network
CSMF	Communication Service Management Function
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
eMBB	Enhanced Mobile Broadband
gNB	Next Generation NodeB (5G base station)
IBN	Intent-Based Networking
IoT	Internet of Things
IRTF	Internet Research Task Force
ISD	Inter-Site Distance
ITU	International Telecommunication Union
LN	Layer Normalization
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
mMTC	Massive Machine-Type Communications
NFV	Network Function Virtualization
NS-3	Network Simulator 3
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnet Management Function
PyBind11	Python Bindings for C++
QoE	Quality of Experience
QoS	Quality of Service
QCI	QoS Class Identifier
RAN	Radio Access Network
RL	Reinforcement Learning
SCS	Subcarrier Spacing
SDN	Software-Defined Networking
SLA	Service Level Agreement
TDD	Time Division Duplex
UE	User Equipment
UMi	Urban Microcell (3GPP channel model)
URLLC	Ultra-Reliable Low-Latency Communications
VNF	Virtual Network Function

List of acronyms used throughout the thesis.

To my family, for their unwavering love and support
and to curiosity which continues to drive me forward.

Chapter 1

Introduction

5th generation (5G) mobile networks are envisioned to support a plethora of services and vertical industries, each with distinct performance requirements. The International Telecommunication Union[1] (ITU) has classified 5G use scenarios into three main categories: enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and massive Machine-Type Communications (mMTC). For instance, eMBB applications such as high-definition streaming of media and virtual reality demand very high data rates, while URLLC services like autonomous driving or industrial automation require guaranteed ultra-low latency and high reliability. The third use case, mMTC, describes scenarios such as the Internet of Things (IoT), involving massive connectivity of low-power devices. These use cases have been defined to accommodate the unique Quality of Service (QoS) needs of different User Equipment (UE) included in 5G networks. The standards that set the stage for 5G, including 3GPP[2], ITU and others, have used these three categories to seek more flexible architectures for 5G that can simultaneously meet diverse service requirements on a common infrastructure [1, 2]. One of the most important technologies introduced with 5G networks is network slicing, a solution that allows a single physical network to be partitioned into multiple virtual end-to-end networks, called slices, which are customized for different services or users. Each slice behaves as an independent logical network with its own configurations and dedicated resources tailored to specific requirements [2, 3]. Using network slicing, an operator can deploy distinct slices for eMBB, URLLC, and massive IoT on the same 5G infrastructure without service interference between them. This concept was first outlined by industry alliances and later standardized by 3GPP to transform the static "one-size-fits-all" paradigm into a dynamic model where multiple isolated slices share the physical network efficiently. Many surveys in both academia and industry have analyzed network slicing and recognized its key enabling role in fulfilling 5G's promise of differentiated services [3, 4]. However, realizing the full potential of

[1] http://itu.int/en/Pages/default.aspx
[2] https://www.3gpp.org/

network slicing in practice introduces new challenges, particularly in dynamic resource allocation. Each slice must receive sufficient resources to meet its Service Level Agreements (SLAs), which may include throughput, latency, and other requirements. Managing this allocation is difficult, as resources are limited and ensuring isolation between slices—while flexibly sharing resources—is a complex task, especially in a fluctuating mobile traffic environment [5]. In the following subsections of the introduction, the 5G service landscape will be discussed, followed by the network slicing paradigm, its associated challenges, and an intent-based networking approach leveraging machine learning as a promising solution.

1.1 5G Service Landscape

5G was introduced to enable a broad range of new services across sectors that greatly exceed the capabilities of previous generations. Earlier in this work the three main service categories of 5G were mentioned, eMBB, URLLC and mMTC. EMBB corresponds to cases requiring extremely high data rates and capacity such as HD video streaming, immersive augmented reality and others. URLLC targets mission-critical services including autonomous vehicles, smart factory controls, remote surgery and power grid protection, which demand end-to-end latencies on the order of milliseconds and near-perfect reliability. MMTC on the other hand focuses on connecting massive amounts of devices (mainly IoT) that individually transmit small amounts of data but in aggregate impose scalability and connectivity challenges. These three categories serve as benchmarks highlighting the heterogeneity of 5G service requirements. The key factor that makes these sets of requirements difficult to meet is that they are expected to be served simultaneously over the same physical network infrastructure. This means that a single 5G network may cater to smartphone users demanding eMBB throughput in the gigabits, industrial robots needing sub-10ms URLLC links and city-wide sensor networks with mMTC connectivity all in parallel. 5G services are used in areas spanning from entertainment to smart cities, industrial IoT and beyond [6]. In an urban city environment, a 5G network may support consumer mobile broadband, vehicle V2X communications and massive sensor grids for environmental monitoring, each with different needs [6]. In the same fashion, a smart factory under industry 4.0 guidelines must rely on 5G to receive reliable low-latency control signals for robotics while also being provided with high-bandwidth links for video surveillance and support for a massive number of wireless sensors. Meeting such varied demands on a common network requires an architectural shift towards flexibility and programmability and this is why slicing was promoted as a solution to accommodate diverse use cases. In essence, the 5G era brings a paradigm where the network must adapt on demand to the service being delivered, rather than adopting a fixed configuration that fits all services poorly.

1.2 Network Slicing and Dynamic Resource Allocation

The heterogeneity challenge faced by the architecture of 5G networks is addressed by network slicing. This novel network architecture logically isolates network resources into multiple slices, each optimized for a particular service. A network slice is defined as an end-to-end logical network spanning the Radio Access Network (RAN), transport, and core network domains, with a specific SLA and resource allocation [2, 3]. The slice abstraction allows operators to provide UEs with customized virtual networks. One slice for instance may be configured to always prioritize low latency and jitter for URLLC, while another maximizes throughput for broadband video streaming. These separate slices operating on the same network share the physical infrastructure but remain isolated in terms of performance and security. 3GPP specifically states in standards, each slice has its own slice ID, dedicated or shared network functions and resource quota [2]. The concept thus decouples service-specific requirements from the underlying hardware, enabling network-as-a-service offerings where new services can be deployed as on-demand slices rather than requiring dedicated physical networks. The structure provided by network slicing may indeed aid with supporting different 5G services, it does not however solve the problem of resource allocation. A critical research question arises from the introduction of network slicing, how should the network dynamically allocate resources to each slice so that all slices meet their SLAs without over-provisioning? This task is particularly challenging as the needs of slices change non predictively with events, emergency situations etc. Simply isolating slices with fixed resource partitions could lead to resources remaining idle in a slice while others struggle to meet their targets with their assigned ones. However, trying to dynamically share resources between slices largely complicates slice management as the system must ensure that one slice's burst in usage does not violate the guarantees of another slice. After these considerations, a slice's lifecycle can be described as involving creation, real-time monitoring, scaling and eventual deactivation. This real time coordination between slices across RAN, core and transport presents a difficult hurdle to overcome, especially as the number of slices increases. Approaching this issue using traditional manual methods is not going to be successful as configuring and tuning each slice by hand is time-consuming and error prone. Researchers in [5] note that generating configuration parameters for every new slice manually not only incurs high operational effort but also cannot react fast enough to the drastic changes in network conditions that occur in live networks. Therefore, automated and intelligent resource management is mandatory for network slicing [5, 7]. The goal of network slicing in modern 5G networks can be described as a slice resource allocation mechanism that can dynamically adjust how resources are distributed among slices based on current network conditions and expressed demands while maintaining isolation and SLA arguments of each slice. Researchers have been exploring solutions to the dynamic slice resource allocation problem and have centered efforts around a promising avenue, the use of Artificial Intelligence (AI) and Machine Learning (ML) to drive closed-loop automation in slice management [4, 7]. An

AI-driven system does not rely on static configuration and heuristics but monitors each slice's performance in real time and learns to make optimized decisions such as reallocating bandwidth and tuning scheduling parameters to meet intents. It has been demonstrated in recent works that supervised learning and deep learning models can predict traffic patterns for each slice and trigger proactive reconfiguration of resources before congestion occurs [8]. In relevant work, a Deep Neural Network (DNN) was used to classify incoming flows by application type and user requirements. This classification enabled the network to automatically direct each flow to the appropriate slice and adjust allocations accordingly which improved efficiency in end-to-end slice management [9]. Other researchers have applied unsupervised learning for anomaly detection in slice performance, or ensemble learning to assist admission control for slicing multiple services [4]. One of the most interesting areas of AI is Reinforcement Learning (RL) with its ability to handle sequential decision-making in complex environments. In the context of network slicing, Deep Reinforcement Learning (DRL) agents can observe the state of the network, e.g. slice load and QoS metrics and act. This may include allocating more resources to one slice or migrating a function to edge cloud for another with the aim of maximizing long-term reward such as overall resource utilization or SLA satisfaction rate. Other research materials report that RL-based approaches outperform static or rule-based policies in dynamic slicing scenarios [6, 10]. In [10], the researchers developed a Deep Q-Network (DQN) agent for 5G RAN slicing that learns how to distribute wireless resources between slices (eMBB vs URLLC) to meet their respective targets. Furthermore, to accelerate convergence, they exploit transfer learning so that knowledge from managing one scenario can be reused when a new scenario is introduced, reducing the training time needed. The adaptive nature of the algorithms used in RL allows them to continuously self-improve by interacting with the network and handle the uncertainty in slice resource allocation better than human-designed policies. In summary, early research results are promising, ML-based slice management can significantly enhance the efficiency of network slicing operations [4, 7, 10]. However, deploying these solutions in operational networks requires a management framework that can interpret high-level requirements and safely delegate control to AI agents. This is where Intent-Based Networking becomes highly relevant.

1.3 Intent-Based Networking and Machine Learning

One of the most recent advancements in network technology is Intent-Based Networking (IBN). This is an emerging area that aims to simplify and automate network management by allowing operators to specify what the network should achieve rather than how to achieve it in terms of device-level configurations. A network administrator in an IBN system expresses high-level objectives to the system, e.g. strict latency below 5ms and the platform automatically translates this intent into network policies and configurations to allocate the proper

resources to fulfill them [6, 11]. Industry leaders and research groups such as IRTF NMRG[3] have added this translation layer on top of Software-Defined Networks (SDNs). IBM envisions IBN as a steppingstone to autonomous networks where human intents are fed to the system, which then self-configures, self-optimizes and self-heals the network to continuously align with those intents. IBN promises to drastically reduce operational complexity in networks with programmable infrastructure such as 5G by abstracting low-level commands. This means that no operator will need to tweak dozen parameters on each network element but instead, he will state the desired outcome and the IBN system will achieve it through closed-loop control and automation. Notably, 3GPP's management architecture for slicing using the Communication Service Management Function (CSMF) aligns with the intent-based concept and researchers have proposed IBN frameworks that integrate with 3GPP slice lifecycle management functions (CSMF, NSMF, NSSMF) to make slicing truly zero-touch [5, 6].

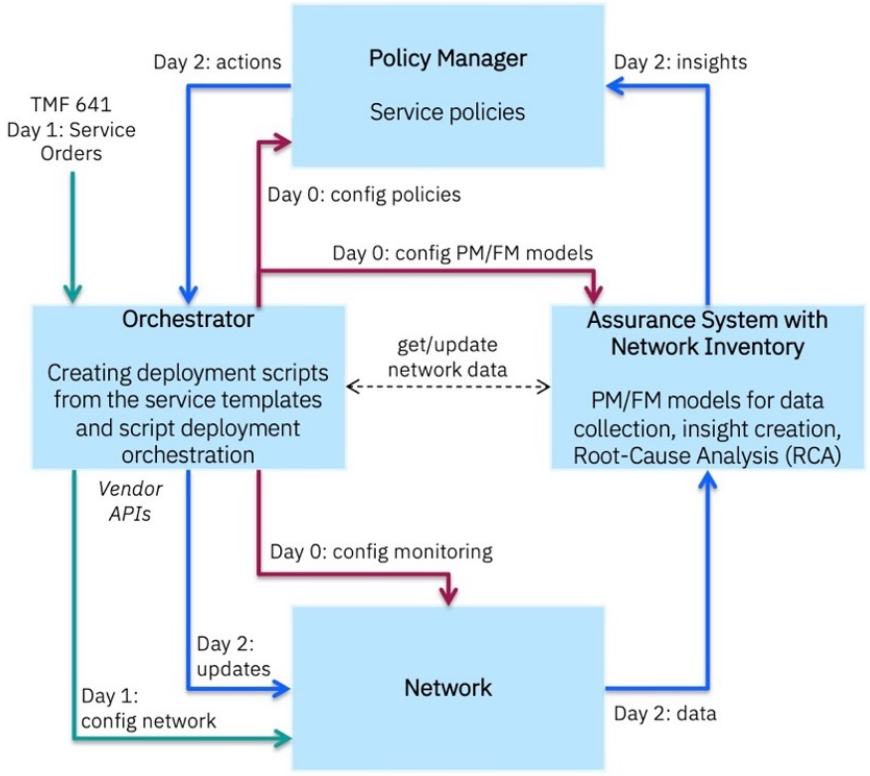


FIGURE 1.1: High-level view of Intent-Based Networking (IBN) architecture and its integration with network slicing and automation.

Industry developments have shown that such a solution may hold merit. One such industry member is Ericsson, offering a Dynamic Network Slicing solution that uses an intent-based service orchestration system to create, modify and assure network slices on-demand, with

[3] https://www.irtf.org/nmrg.html

automated closed-loop adjustments based on real-time analytics [12]. This system allows operators to define slicing policies at a high level such as by service type or enterprise customer needs and the platform takes care of instantiating the necessary Virtual Network Functions (VNFs) and continuously controlling network resource partitions to uphold those policies. In early trials, this approach has shown network reaction times close to milliseconds which is faster than a human network operator. It is indicated therefore, that IBN paired with AI/ML is a promising solution to the dynamic resource allocation problem in 5G and beyond where high-level intents guide the objectives and ML agents handle the low-level decision-making. In conclusion, a cooperative IBN and ML approach may bridge the gap between business goals and network configurations, enabling 5G network slices to be managed in an autonomous and closed-loop manner that ensures each slice gets the right resources at the right time to meet its SLAs. This thesis building upon the established concepts will investigate an architecture that leverages IBN to interface with a Deep Q-Network agent for slice resource management and evaluate its effectiveness in orchestrating 5G network slices under dynamic conditions.

1.4 Semantic Communication in 5G

Traditional communications systems focus on the accurate delivery of bits, however, an emerging field of communications is semantic communications, which emphasize on the meaning of the transmission. In this approach, the goal is not simply to minimize bit errors but to ensure that the received message conveys the intended knowledge and fulfills a higher-level task objective. This field of communications incorporates elements such as context, intent and task relevance in to the communication process to align the network operation more closely with user goals. For instance, in an mMTC scenario the age of information may be more valuable than the raw data rate, whereas in URLLC, the semantic success of a control command may outweigh the delivery of redundant packets. By integrating semantic metrics into network control, future networks can achieve higher efficiency, since resources are allocated to maximize the fullfilment of service objectives rather than just raw performance. This work adopts such a semantic-aware perspective, extending IBN by embedding semantic indicators directly into the decision-making of the Deep Reinforcement Learning (DRL) agent for 5G slice management.

Chapter 2

Literature Review

The transition of mobile communication systems to the 5th generation (5G) represents a pivotal technological advancement that enables ultra-reliable low-latency communication (URLLC), enhanced mobile broadband (eMBB) and massive machine-type communication (mMTC). These service classes underpin a wide variety of applications that range from industrial automation and real-time healthcare to immersive media and smart cities. Each one of these use cases comes with distinct service-level agreement (SLA) requirements such as specific latency, reliability and throughput. To accommodate such heterogenous demands inside a unified structure, network slicing has emerged as a fundamental architectural concept, allowing physical network resources to be logically partitioned into isolated virtual networks that operate in a service-specific context [13]. Network slicing is implemented both at the Radio Access Network (RAN) and core domains, often in conjunction with Software Defined Networking (SDN) and Network Function Virtualization (NFV). To ensure SLA compliance, the orchestration of slice resources in a dynamic manner is vital. However, network slicing introduces new complexity in policy enforcement and quality of service (QoS) differentiation. For instance, slices intended for URLLC must guarantee sub-millisecond latency and near-zero packet loss, requirements that are difficult to fulfill in high-mobility or congested environments [14]. Consequently, managing slice behavior has become a central challenge in the operation of 5G systems. In recent studies, deployments have revealed practical bottlenecks in real-world 5G networks. In particular, performance evaluations in campus environments indicate that despite the theoretical support for slicing, current infrastructure struggles with effective SLA isolation and mobility-related disruptions. As proven by the results of a university-scale 5G network measurement study, significant handover delays and inconsistent throughput were observed across different service types, especially under high user load and urban mobility scenarios [15]. This fact highlights the importance of advanced control techniques that can adaptively orchestrate network slices with intelligent decisions based on real-time context. Furthermore, industry verticals such as smart manufacturing and autonomous vehicles represent one of the

most demanding testbeds for slicing effectiveness. In industrial scenarios, system designers face stringent requirements on latency, jitter and reliability. The slicing-enabled RAN must support isolated scheduling, traffic shaping and failure recovery, many times using customized radio resource management (RRM) policies. Researchers focusing on 5G RAN slicing in such settings have proven that while static slicing policies provide baseline QoS guarantees, they fall short under dynamic workloads or unexpected traffic spikes [15]. This further motivates the integration of intelligent control mechanisms to adaptively tune slice configurations in real time. In the future 6th generation (6G) systems, network slicing is expected to evolve into more flexible and autonomously managed paradigms. The global 6G landscape is shown in recent comparative analyses to present a tendency to move from predefined slices to on-demand slice instantiation driven by semantic information and user intent [16]. This anticipation will require more than just virtualization, it creates a demand for a radical rethinking of network management using artificial intelligence, semantic encoding and intent-based frameworks. It is evident that while 5G introduces the mechanisms for service differentiation via slicing, real-world deployments present gaps in their optimization and automation. The growing complexity of application demands especially in mobile and heterogenous environments call for intelligent and policy-driven solutions. These foundations justify the need for advanced techniques such as intent-based networking, semantic communication and deep reinforcement learning which will be explored in the following sections. The evolution of network management from manual configuration to autonomous control has underscored the emergence of Intent-Based Networking (IBN) as a transformative architectural principle. IBN allows network operators to express high-level objectives called intents in human-understandable form, which are then automatically translated into network policies that are enforced in real time. This is an important asset utilized in 5G and future 6G environments, where differentiated services must be delivered over dynamic and varying infrastructures [17]. The architectural blueprint for IBN evolves around four main phases that include intent translation, orchestration, monitoring, and closed-loop assurance. In 5G/6G networks, the complexity of intents that ranges from performance constraints to service prioritization creates the need for intelligent mechanisms that can interpret user demands and translate them into enforceable configurations. One prominent direction in realizing this vision is the integration of Graph Neural Networks (GNNs) into IBN frameworks. In a recent study, a model in which IBN is used for dynamic and efficient path selection based on SLA requirements was introduced [18]. In this system, GNNs are leveraged to analyze topological characteristics and predict optimal routing paths in real time. Unlike classical shortest path algorithms, the GNN can process variable-length input graphs, making it suitable for dynamically changing network conditions. The researchers demonstrated up to 30 % improvement in SLA satisfaction under congestion, with GNNs enabling rapid adaptation to network events. To further engage GNNs with IBN, other studies have suggested a congestion-aware routing agent for IBN systems [19]. This approach differs from the previous discussed in its objective, rather than focusing optimization efforts to latency, the model seeks to maximize

user Quality of Experience (QoE) across heterogeneous services. To achieve this, a multi-head GNN is trained to classify congestion severity and recommend rerouting strategies. Following simulations, results showed significant performance improvements in VoIP and video streaming services, confirming the utility of GNNs in slice-aware and QoS-sensitive routing decisions. Moreover, other research materials explore cross-domain orchestration using IBN for Space-Air-Ground Integrated Networks (SAGIN) [20]. The proposed framework integrates GNNs with Deep Reinforcement Learning (DRL) to jointly manage resources across terrestrial, aerial and satellite domains. The DRL agent observes environmental metrics such as link stability, coverage and interference while GNNs assist in learning inter-domain correlations. This complete architecture interprets user intents such as "maximize video stream stability during drone flight" and maps them to a coordinated resource allocation policy. Compared to baseline methods, the system improved resource utilization by 25 % and reduced latency by 40 %. Research efforts have also explored methods to simplify the expression of intents, with Natural Language Processing (NLP) and Large Language Models (LLMs) appearing as promising solutions to create front-end enablers for IBN. In such a study, contributors developed an NLP-powered IBN system that supports natural language commands including "ensure 20 ms latency for surgical robot slice" [21]. The solution employs a transformer-based encoder-decoder model that translates these expressions into structured policy graphs. Then, the backend enforces these using an SDN/NFV orchestrator. Real-world testing on private 5G setups shows that such NLP interfaces reduce configuration time by 60 % and minimize intent ambiguity. A more complete approach was presented in [22], where Maestro, a collaborative LLM-driven IBN orchestrator for 6G networks was introduced. Maestro interacts with large-scale language models such as GPT and BERT to parse and refine user intents, even in ambiguous cases. The system employs multi-agent cooperation to verify and simulate intent outcomes before deployment. In high-mobility vehicular network scenarios, Maestro achieved adaptive reconfiguration latency under 1 second and reduced SLA violations by 45 % compared to static policy systems. This showcases the promise of LLMs in democratizing access to complex network automation tools. To make use of the interpreted intents, robust closed-loop control must be used to ensure enforcement. Researchers have proposed a closed-loop IBN architecture for Open RAN environments [23]. This solution combines real-time telemetry collection, KPI assessment and automated remediation. For instance, if an intent requests "minimize URLLC jitter below 5 ms" the system monitors link delay, detects violations and proactively shifts resource blocks or applies redundancy. The included experimental testbed demonstrates stable SLA enforcement even under fluctuating radio conditions. This concept was further expanded by [24], where a GNN-based system anticipates potential SLA violations using historical and contextual data, introducing predictive IBN agents. Prediction intervals of up to 30 seconds enable early intervention, such as preemptive resource migration or path adjustment. In a simulated 5G transport network, this reduced violation frequency by 50 % and improved energy efficiency via smarter link activation. A holistic IBN architecture with AI-native components is

demonstrated with the NEMI framework [25]. Designed for open campus networks, NEMI supports many features, including intent registration, semantic parsing, SLA policy resolution and execution. The engineering behind it is modular, which allows integration with SDN controllers and service orchestrators. Notably, NEMI includes a Semantic Processing Unit (SPU) that performs hierarchical decomposition of intents, e.g. "guarantee eMBB throughput in zone B" becomes a chain of bandwidth, priority and mobility sub-intents. The system supports 6G use cases such as slice-on-demand and cross-domain automation. One more crucial element for production-grade IBN systems is provenance. It is argued by researchers that IBN should include an audit trail mechanism that tracks the lifecycle of each intent, including submission, translation, application and retraction [26]. The proposed system attaches a unique identifier and metadata to every intent object, enabling backtracking in case of policy conflicts or compliance issues. This feature is particularly critical in domains like healthcare where SLA breaches may have legal ramifications. In similar fashion, a framework called NEMI is presented as a 6G-ready autonomic manager [25]. Beyond intent translation and orchestration, NEMI includes an autonomic loop that dynamically adapts to user behavior. For instance, during a large-scale event, the system can infer from demand patterns that slice reassignment or cell splitting is needed, even without explicit intent refresh. This intelligence is key to effectively support self-driving networks where minimal human intervention is expected. In summary, the IBN ecosystem has progressed from simple intent-to-policy translation to fully autonomous, AI-driven, predictive orchestration, with GNNs enhancing structural reasoning, LLMs democratizing access and DRL enabling adaptive enforcement. Nevertheless, most implementations operate in controlled testbeds or lack integration with real-time simulators like NS-3 . Furthermore, they often focus on either control or communication layers, without addressing semantic understanding from real-time feedback and cross-layer optimization in a unified framework. This thesis proposes to bridge these gaps by integrating IBN, semantic communication and DQN-based optimization in a real-time NS-3-based simulation environment with shared memory interaction, offering both architectural novelty and practical reproducibility. In the following years, the transition towards 6G is planned and communication systems face a paradigm shift, not just in speed and capacity but in intelligence. Communications till now were built on Shannon's theory, with a focus on accurate bit-level transmission without regard to the meaning of information. However, in scenarios such as the one studied, focus on emergency response, it is not sufficient to merely transmit bits but what matters is what the receiver needs to understand. This emerging concept is known as semantic communications and it is poised to become foundational in 6G architectures [27]. This technology seeks to transmit meaning rather than raw data. The content of the information refers to the actionable knowledge embedded in the transmitted message, such as the intent behind an emergency alert or the behavioral context of a user device. This approach as shown in literature enables a reduction in the required bandwidth, improve latency performance and higher relevance of transmitted content [28]. The main hierarchical levels of communication in 6G are classified as syntactic

(bit-level), semantic (meaning), and effectiveness level, meaning how useful the message is for the recipient's task, with the semantic and effectiveness layers gaining prominence. One of the most innovative aspects of this shift is semantic decomposition, where source data is split into core semantics and contextual layers, allowing for prioritized transmission. Researchers have further suggested a scalable semantic coding framework that separates messages into hierarchical semantic units [29]. Each layer is encoded according to its importance for the receiver's inference task. Experimental results demonstrate that semantic-aware transmission reduces bandwidth by up to 40 % without degrading task performance. This is a very promising path to achieve a better latency score, especially when URLLC is a requirement. In practice, semantics are put to the test with emerging architectural frameworks. In other work, the authors design a Hybrid Semantic Radio Access Network (RAN) protocol stack in which semantic layers are introduced between traditional PHY/MAC and network layers to enable real-time prioritization of semantically critical packets [30]. The system was developed using a deep neural inference module in a 6G simulator, showcasing significant throughput improvements in vehicular and mission-critical environments. This approach bridges physical-layer innovations with semantic policy enforcement a fact that aligns closely with the objectives of this thesis. Further work in [31] showcases the role of semantics in Machine-Type Communication (MTC) where six key features are outlined. Some of those include intelligence, adaptiveness, and energy efficiency which are all supported by semantic communication. The authors of this work argue that semantic-aware scheduling enables efficient support for massive and heterogenous IoT devices. With this approach, rather than transmitting periodic updates blindly, devices can assess whether the semantic content has changed meaningfully before initiating communication. This reduces signaling load and improves scalability. A collective outlook of these developments underscores that semantic communication is not simply a new encoding technique but a cross-layer design philosophy that redefines how communication systems interpret and act on information. By transmitting "what matters" rather than "what is said", networks can become more efficient, responsive and intelligent. This is essential in situations where time and relevance are critical, such as emergency response and autonomous control. The mentioned materials seem promising, especially in the planned move towards 6th generation networks, however, the integration of semantic communication into real-time network control remains an open research challenge. Most systems are either theoretical, simulation-based, or confined to inference tasks. Furthermore, the tight coupling between semantic meaning and network intent, e.g. prioritizing emergency semantic content in URLLC slices is still mostly unexplored. This thesis aims to address this intersection by incorporating a semantic-aware layer into an IBN-driven NS-3 simulation, where decisions on slice scheduling and resource allocation are based not only on QoS metrics but also on semantic importance, dynamically interpreted by a DQN agent. Network slicing is one of the key enablers of 5G and 6G networks that allows for a single physical infrastructure to support multiple logical networks

tailored to different service requirements. Slices can be provisioned for latency-critical services such as URLLC, bandwidth-intensive services like eMBB or other applications. However, static slicing strategies are not enough in highly dynamic environments. This has prompted a shift toward AI-driven and intent-based orchestration strategies, where resource management is continuously adapted using learning models. In this section, several key works are analyzed that demonstrate the application of different technologies in network slicing to pave the way toward fully autonomous and semantically aware networks. Similarly, the work presented in this thesis will feature an NS-3 simulation interacting with a Deep Q-Network (DQN) agent to schedule slices dynamically depending on semantic urgency and link conditions. Researchers in [32] propose a dynamic framework for Machine Learning algorithm selection tailored to network slicing across Beyond 5G (B5G) domains. Rather than relying on a fixed model, their system evaluates current network states, service demands and slice KPIs to dynamically choose between supervised, unsupervised and Reinforcement Learning models. For instance, when latency-sensitive applications dominate, the framework selects RL agents to ensure timely feedback adaptation. The included experiments show that switching algorithms at runtime reduces SLA violations by 23 % and improves slice-level resource utilization by 35 %. Further work in [33] provides a broad taxonomy of ML/DL models for 5G slicing, organizing use cases into three categories: (1) slice admission, (2) slice placement, and (3) slice adaptation. The study emphasizes that Deep Reinforcement Learning (DRL), particularly Deep Q-Networks is most effective when slice allocation needs to consider high-dimensional state spaces and QoE metrics. Using real operator traces, the authors show that DQNs adapt more effectively to fluctuating traffic loads and outperform static resource managers by up to 19 % in throughput efficiency. This supports the architectural decision in this thesis to utilize a DQN agent that dynamically assigns bandwidth across URLLC, eMBB and best-effort slices by monitoring feedback from the simulation. As described in 5G standards releases, the reliability requirements of industrial communication systems, such as factory automation or healthcare monitoring, impose stringent QoS constraints. This is why researchers in [14] develop a slice-aware scheduler deployed in a real 5G Radio Access Network (RAN). Their system isolates latency-sensitive traffic into dedicated slices and continuously monitors application-layer KPIs to adjust prioritization. As reported from their findings, their testbed shows >99.99 % reliability when the scheduler incorporates real-time service feedback. This demonstrates the feasibility of cross-layer control, which this work extends by introducing semantic awareness into the slicing policy. One of the more future focused works is the 6G-LEGO framework introduced in [34] that reimagines slicing in 6G as a modular AI-native service composition. Each slice is constructed from "semantic bricks," i.e., functionally decomposed and reusable AI agents or services. The system leverages multi-agent deep reinforcement learning to compose and adapt slices in real time, with policies aligned to network intents. Simulation results show up to 40 % reduction in orchestration delay and 25 % fewer SLA violations in mobile networks. While this is a theoretical framework, its focus on semantic alignment and intent-aware orchestration is

in alignment with the scope of our proposed framework. Slicing in mobile environments introduces the challenge of maintaining service continuity across handovers. It is shown in study [35] that this hurdle can be overcome with a handover-aware slicing mechanism designed for a specific use case, such as for high-speed mobility where frequent base station switching can degrade service. Their method uses graph-based trajectory prediction to pre-allocate resources and reduce handover events by up to 50 %, while maintaining latency bounds. This is further validated with [36] where a 283 km/h field trial using the Shinkansen train was completed. In this experimental setup, a 5G system using beamforming and predictive slicing achieved low-latency, high-throughput service despite rapid movement. Their results underscore that slicing must incorporate predictive models and mobility state inputs, supporting the idea that deep learning-based scheduling in realistic mobility conditions is both necessary and achievable. In order to transition from simulation to deployment, it is required to create real testbeds that integrate AI into the 5G stack. In [37], researchers present a virtualized testbed combining OpenAirInterface, Docker-based VNFs and AI-driven slicing components. The system supports semantic-based traffic classification and slice assignment using TensorFlow models with the key innovation being the closed-loop feedback between semantic detection and slicing, with intelligent routers steering traffic into contextually relevant paths (e.g., routing medical images over URLLC slices). Their modular setup offers a practical foundation for evaluating semantic-aware slicing in real environments. The works presented in this section establish that AI-driven slicing is central to network automation, especially in heterogeneous and mission-critical contexts. They also demonstrate that Depp Q-networks are well-suited for dynamic slicing under uncertainty and semantic information can guide more efficient allocation. In recent works there have been attempts to bridge the gap between simulation and deployment, however several steps need to be taken to reach this goal. It is also established that mobility-aware slicing is essential in realistic topologies where there are multiple users moving across different base stations and different coverage areas. Despite progress, few implementations combine semantic communications with DRL slicing control in realistic simulators. This thesis addresses this gap by deploying an NS-3 system integrated via shared memory with a DQN agent trained to assign resources based on intent and semantic value, not just QoS. Semantic Communication has shown to be a promising enabler for next-generation networking with the aim to reduce overhead and enhance goal-oriented transition [27, 28], however, its use in network slice orchestration remains superficial. Implementations discussed in this section like the Hybrid Semantic RAN [29] and semantic decomposition coding methods [28] focus on improving physical or MAC layer efficiency. These implementations though do not incorporate semantic metrics into higher-layer AI-based control or intent-based policies. Moreover, even advanced systems created for 6G such as Maestro [22] and NEMI [25] propose automation via semantics but fall short in dynamically adjusting slice behavior based on semantic content classification. This reveals a significant gap in exploiting semantic context as a slice orchestration variable. As mentioned in recent literature, Deep Reinforcement Learning and specifically

Deep Q-Networks (DQN) are promising approaches for resource allocation [32, 33], but they are often trained and evaluated offline, using static topologies or simplified simulators. For instance, in [32], researchers propose algorithm selection frameworks in Beyond 5G but rely on precomputed performance databases, while in [33] they provide a taxonomy of ML and DL slicing methods without focusing on real-time simulator integration. Additionally, even in testbeds such as [37], while AI agents interact with VNFs, there still is no seamless coupling with NS-3-level simulation, especially using shared memory for quicker data exchange. This lack of cooperating simulation environments limits the capacity to observe dynamic QoS or semantic state changes and learn from environment transitions in real-time. Moreover, it hinders the ability to adjust intent-fulfilling behavior during ongoing sessions. Furthermore, in included studies tackling AI slicing matters, RL agents optimize slices based on classical parameters such as bandwidth, latency or user count. However, there is no evidence of agents trained with semantic flow metadata like message priority, origin intent or emergency classification as part of their state vectors. This is a missed opportunity for context-sensitive decision-making, where network behavior aligns with the meaning or criticality of data. Even in a state-of-the-art architecture like 6G-LEGO [34], which promotes modular slicing, semantic-awareness is not leveraged for dynamic policy adjustment. Even though Machine Learning studies report positive simulation results, they often employ toy models or train on simplistic network environments. For instance, [32] and [33] evaluate ML in static slicing scenarios. In contrast, this work includes realistic features such as hexagonal gNB deployment, variable UE mobility patterns and simultaneous URLLC, eMBB and best-effort flows. In addition, studies like [35] and [36] demonstrate the impact of mobility and beamforming in real-world 5G trials but they do not integrate AI-based slicing. Thus, existing DRL agents remain untested in truly heterogeneous and time-evolving environments.

The following figure presents a three-level model of semantic communication that shows the flow from source to destination, including source/destination knowledge, semantic/syntactic channels and noise/interference.

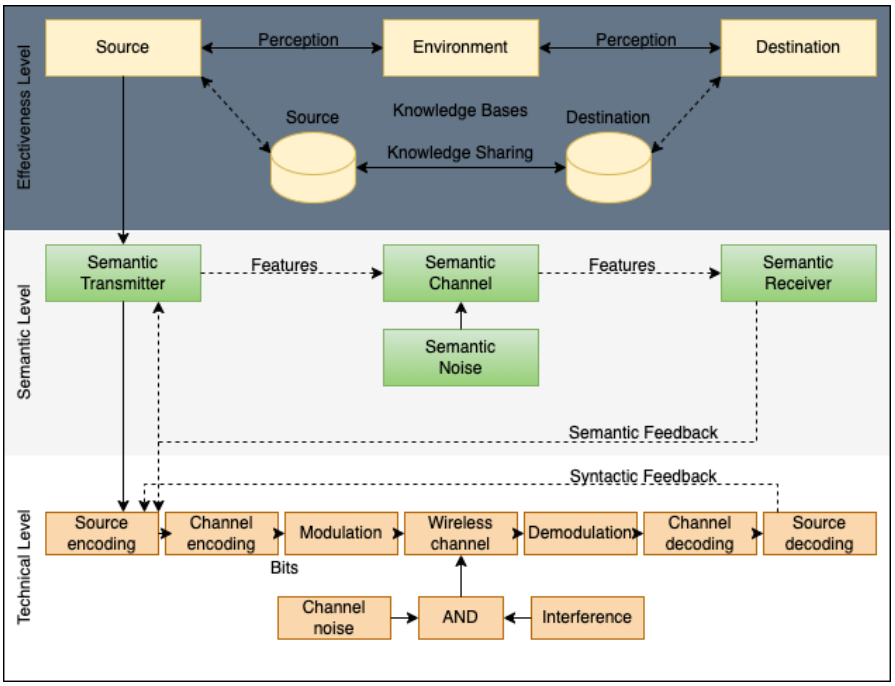


FIGURE 2.1: Three-level model of semantic communication.

All the identified limitations in current research present gaps that need to be addressed. This thesis in response proposes a real-time NS-3 simulation framework integrated with a Python-based Deep Q-Network agent that orchestrates semantic-aware, intent-driven slicing. The framework leverages shared memory and tensor exchange between the simulator and learning agent to enable real-time policy updates. The simulation environment is built upon the official NR module in NS-3 and is used to build a 5G RAN topology that consists of 7 hexagonal gNodeBs and multiple mobile UEs with varying flow types. The slices that are implemented are logically isolated and include a URLLC emergency, an eMBB emergency and a best effort. The DQN agent is implemented in Python and receives tensor-encoded environment states through a shared memory interface. Then, in real time, it returns slice selection decisions. This design directly addresses the static limitations of prior slicing models [32], [33]. The proposed system also introduces a semantic layer that classifies packets and flows into categories such as critical alerts, video telemetry and others. This classification is embedded into the DQN's input state vector, enabling it to prioritize semantically meaningful flows even when bandwidth is limited. In contrast to prior systems like NEMI [25] or 6G-LEGO [34], which lack semantic-driven slice control, this thesis leverages semantics for real-time prioritization logic. The DQN agent learns to serve emergency alerts on URLLC slices while balancing throughput on eMBB slices for multimedia sessions, aligning with the conceptual goals of semantic networking proposed in [28] and [14]. To further boost the realism of this work and put to the test many of the NR module's functionalities, unlike many academic works, this implementation uses varying transmission power according to demands, realistic fading models and hexagonal cell layout and also beamforming. Furthermore, real-time performance metrics are

captured with NS-3's flow monitor and the agent that include slice SLA monitoring, latency, throughput and others. This simulation serves as a practical validation of the autonomous intent-slicing-semantic feedback loop, which is currently theoretical in much of the literature. This thesis offers an innovative approach towards IBN-compatible 5G systems where semantic flow data guides slice prioritization and DRL agents adapt in real-time to intent policies. The shared memory-based architecture of the simulation enables co-evolution of network and intelligence layers. It builds a conceptual and technical bridge between the semantic communication paradigm, intent-driven networking architectures and ML-based 5G slicing. Moreover, it demonstrates their unified viability in an emulated 5G environment. As 6G research accelerates toward autonomous and meaning-driven networks, this thesis lays the groundwork for deployable AI-native network management systems.

Chapter 3

Notation & Fundamentals

In this chapter, the technical foundation of this thesis is established and the basic elements of the architecture are presented. Furthermore, the mathematical notation and state definitions are showcased and the core principles of Deep Reinforcement Learning, Intent-Based Networking and semantic communication are introduced.

3.1 Notation Table

Table 3.1: Notation Table

Symbol	Description
$\mathcal{S}$	Set of all possible environment states. Each state represents the system's status such as traffic load, latency, semantic relevance and slice utilization.
s_t	System state at time step t as observed from the NS-3 simulation environment.
f_t	Feature vector derived from state s_t that incorporates metrics such as semantic entropy, priority and encoded context.
$\mathcal{A}$	Action space representing all feasible decisions available to the agent (e.g., slice assignment, bandwidth allocation).
a_t	Action selected at time t based on the policy derived from the Q-network.
r_t	Reward received at time t, calculated from QoS metrics such as SLA satisfaction, throughput and semantic accuracy.

Continued on next page

Table 3.1 – *Continued from previous page*

Symbol	Description
γ	Discount factor for future rewards. It controls the importance of long-term vs immediate rewards in DQN learning ($0 < \gamma \leq 1$).
ϵ	Exploration rate used in the ϵ-greedy action selection policy to balance exploration and exploitation.
α	Learning rate used in the stochastic gradient descent updates of the Q-network. It controls the step size for weight updates.
θ	Trainable weight parameters of the primary Q-network.
θ^-	Parameters of the target Q-network used in Double DQN for stabilized learning.
$Q(f_t, a; \theta)$	Estimated action-value function mapping feature vectors and actions to expected cumulative reward.
ℓ	Temporal Difference (TD) loss function minimized during Q-network training: $\ell = (y_i - Q(f_t, a_t; \theta))^2$.
y_i	TD target computed as: $r_t + \gamma \max_{a'} Q(f_{t+1}, a'; \theta^-)$. It is used to update the Q-network.
$\mathcal{D}$	Experience replay memory buffer that stores tuples $(f_t, a_t, r_t, f_{t+1}, \textit{done})$ for training stability.
τ	Soft update coefficient controlling the periodic update of the target network parameters: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$.
done	Terminal flag indicating whether the current episode has ended.
T_{inf}	Inference time per decision cycle, representing the real-time responsiveness of the agent.
P_{sem}	Semantic priority score associated with a given data packet, used to aid with slice selection.
E_{sem}	Semantic entropy score representing the semantic efficiency of encoded message content.
Φ	Semantic mapping function applied to incoming data to extract latent features or intents.

3.2 Deep Reinforcement Learning and DQN

Reinforcement Learning (**RL**) constitutes a class of Machine Learning methods in which an agent interacts with a dynamic environment and learns to perform actions that maximize cumulative long-term rewards. The decision-making process is modeled as a Markov Decision Process (**MDP**), defined by the tuple $(\mathcal{S},\ \mathcal{A},\ \mathcal{P},\ \mathcal{R},\ \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of possible actions, $\mathcal{P}(s'|s,a)$ denotes the transition probability from state s to s' under action a, $\mathcal{R}$ is the reward function and $\gamma \in [0,1]$ is the discount factor [38].

The optimal policy π^* aims to maximize the expected return defined as:

$$E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{3.1}$$

In environments with high-dimensional and continuous state spaces, traditional tabular methods become intractable. This limitation is addressed by Deep Reinforcement Learning (DRL) by using a deep neural network to approximate the action-value function, known as the Q-function. In the Deep Q-Network (**DQN**) algorithm, the Q-function is represented as $Q(s,a;\theta)$, where θ are the trainable parameters of the neural network.

DQN combines Q-learning with function approximation, experience replay and target network stabilization. The temporal-difference (TD) error, which drives learning, is given by:

$$\delta_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta) \tag{3.2}$$

The loss function minimized during training is:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}} \left[\delta_t^2 \right] \tag{3.3}$$

To decorrelate data and improve stability, transitions are stored in a replay buffer $\mathcal{D}$, from which mini-batches are sampled randomly during training. Moreover, to mitigate the problem of Q-value overestimation, Double DQN introduces the use of two networks: the online network $Q(s,a;\theta)$ and the target network $Q(s,a;\theta^-)$.

The Double DQN update rule modifies the TD target as follows:

$$y_t = r_t + \gamma Q\left(s_{t+1}, \operatorname*{argmax}_{a'} Q(s_{t+1}, a'; \theta); \theta^- \right) \tag{3.4}$$

This mechanism ensures better convergence properties and lower variance in learning, which is especially important in systems with real-time constraints such as 5G RAN slicing.

The use of DRL in 5G networks has been studied to propose optimizations for dynamic resource allocation and network slicing [8, 10]. In this thesis, the DQN agent is responsible for selecting optimal slice assignments (URLLC, eMBB, and BestEffort) in a virtualized RAN setup based on real-time flow level state metrics extracted from NS-3. Each state s_t is represented by a feature vector $f_t \in \mathbb{R}^n$ that includes semantic and physical attributes such as delay, packet loss, flow entropy and priority weights. Actions $a_t \in \mathcal{A}$ represent RAN slicing decisions for flows (UEs per slice). The environment is modeled in NS-3 with 7 gNodeBs and 12 UEs under a hexagonal topology and responds with new states and a reward signal. The reward function is designed to penalize violations of SLA constraints such as latency greater than 5 and 10ms for URLLC, jitter greater than 20ms for eMBB, and reward efficient usage of semantic bandwidth, inspired by recent works on intelligent slice management [5].

To enhance the learning process additionally, a Prioritized Experience Replay (PER) mechanism is used to sample transitions with higher TD error $|\delta_t|$ with higher probability. This ensures that critical learning experiences are emphasized to improve sample efficiency and convergence speed.

The target network is updated via a soft update mechanism:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{3.5}$$

where $\tau \in [0, 1]$ is a small value such as 0.001. This slow update prevents oscillations and ensures stable target values across training iterations.

The proposed DQN agent integrates semantic communication principles. Metrics such as throughput and delay are extended with semantic metrics including message entropy E_{sem}, contextual priority P_{sem} and relevance rate R_{sem}. These values are derived using lightweight parsers and mapping functions from flow-level descriptors, in accordance with frameworks presented earlier [27, 28]. Incorporating semantic indicators into the state vector allows the agent to make decisions that preserve the meaningfulness of transmitted data rather than merely optimizing raw QoS metrics. This aligns with recent trends in semantic-aware slicing where flows with high contextual importance are routed via URLLC slices even under moderate bandwidth constraints.

However, in this implementation, the semantic layer adopts a higher-level abstraction focused on network intent representation through feature encoding. Specifically, flow-level attributes such as latency sensitivity, priority class, and service type are distilled into a semantic feature vector f_t, which encapsulates task-relevant meaning without requiring full payload interpretation. This abstraction enables the DQN agent to learn policies that are aware of service intent,

such as prioritizing **URLLC** over **eMBB** or best-effort slices, even in scenarios with shared radio resources or conflicting QoS demands. This approach follows the trend of semantic-assisted AI agents that use semantics as input priors that improve learning stability and convergence in dynamic environments. This contradicts other works that focus on knowledge graph inference [27] or end-to-end semantic resource encoding [28] and integrates lightweight semantics at the observation layer, ensuring compatibility with real-time constraints and NS-3's simulation stack. The approach is similar in spirit to the architectural layering suggested in [30], where semantic abstraction occurs at the edge and feeds into the learning model without altering the packet-level stack. It is evident that the semantic module in this case does not perform explicit symbol-level reasoning but rather infers user or service intent from abstracted flow features, forming the bridge between traditional observation vectors and intent-driven DQN action policies. This integration allows the agent to operate in a hybrid Intent-Based Networking (IBN) paradigm where semantic features enhance context-awareness and decision accuracy without incurring excessive computational overhead.

One further innovation that is a part of this thesis implementation is the real-time interaction between the Python DQN agent and the NS-3 C++ simulation via shared memory buffers. At each simulation timestep, NS-3 exports flow-level statistics into a POSIX shared memory array, which is parsed by the agent using the `mmap` interface. The agent then preprocesses the vector f_t, selects an action a_t, and writes the decision back into the shared memory where the NS-3 controller thread applies the new slicing policy. This architecture allows continuous control with less latency, enabling real-time adaptability to flow changes which remains largely underexplored in literature. Unlike offline optimization or periodic control, the proposed agent executes a closed feedback loop with the network emulation, ensuring responsiveness and learning under non-stationary traffic scenarios.

Algorithm 1 Semantic NR-Slicing Simulation Environment Loop (**NS-3 Side**)

Initialize: 5G NR network topology, traffic generators, semantic KPIs
Start shared memory interface (`ns3ai_nr_slicing_msg`) for agent communication
for each simulation time step t **do**
    Measure per-slice throughput, latency, and resource utilization
    // **Compute Semantic Layer Features**
    Compute:
    - URLLC Semantic Success
    - eMBB Semantic Ratio
    - mMTC Age of Information
    - Intent Satisfaction
    - QoE for each slice
    - Network Efficiency, Congestion, SLA Compliance
    Form state vector s_t (core + semantic features)
    Send s_t to agent via shared memory
    Wait for agent action a_t (allocation vector)
    // **Apply Agent Action**
    Decode a_t to slice resource fractions
    Update scheduler/resource allocation for URLLC, eMBB, mMTC
    Advance simulation by Δt
    if simulation end **then**
        Send `done` signal to agent and exit loop
    end if
end for

Algorithm 2 Double DQN Agent for Semantic-Aware 5G NR Slicing (**Python Side**)

Initialize: Q-network with parameters θ, target network θ^-, prioritized replay buffer $\mathcal{D}$
Set ϵ (exploration), learning rate α, discount γ, soft update τ
while training not complete **do**
    Receive state s_t from environment (semantic vector)
    // **Action Selection with Heuristics**
    if random() $< \epsilon$ **then**
        Select a_t randomly or use heuristic (e.g., URLLC/emBB focus)
    else
        $a_t \leftarrow \arg\max_a Q(s_t, a; \theta)$
    end if
    Send a_t (allocations) to environment
    Observe reward r_t, next state s_{t+1}
    Store $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in $\mathcal{D}$
    // **Experience Replay with Prioritization**
    if training step **then**
        Sample mini-batch from $\mathcal{D}$
        for each sample $(s_i, a_i, r_i, s_{i+1}, \text{done}_i)$ **do**
            if done_i **then**
                $y_i \leftarrow r_i$
            else
                $y_i \leftarrow r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta^-)$
            end if
        end for
        Compute loss $\ell = \sum (y_i - Q(s_i, a_i; \theta))^2$
        Update θ via AdamW optimizer
        Soft update: $\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$
    end if
    Decay ϵ with annealing
    if termination or simulation end **then**
        Save model and results, exit loop
    end if
end while

In the following figure, the co-simulation flow can be seen, with the NS-3 simulator and semantic-aware Python DQN agent interacting in real time via a shared memory interface. Resource allocation decisions are made by the agent and applied to the simulation, with performance metrics and semantic indicators exchanged every step.

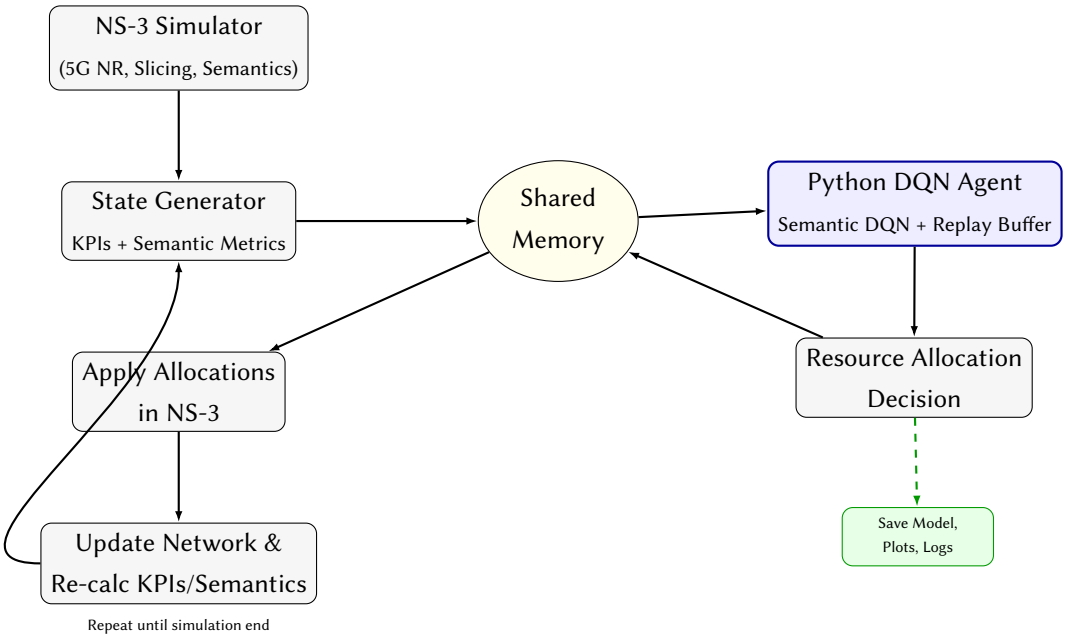


FIGURE 3.1: Co-simulation flow of the system.

3.3 Intent-Based Networking (IBN) Integration

Intent-Based Networking (IBN) represents a transformative paradigm in network management, wherein high-level service or business intents are dynamically translated into actionable network configurations and resource allocations [6]. In this 5G network slicing framework, IBN provides an abstraction layer that enables dynamic, context-aware mapping of intents (such as ultra-reliable low latency for emergency traffic or enhanced throughput for multimedia) into concrete slice resource allocations and scheduling priorities.

System Implementation: The simulation architecture integrates IBN through a closed-loop with a Deep Q-Network (DQN) agent. High-level intents, corresponding to slice-specific service requirements, are expressed as context-driven allocation strategies. The NS-3 environment continuously monitors slice performance, capturing both classical metrics (throughput, latency, packet loss) and an expanded suite of semantic communication indicators, including semantic success, age of information, Quality of Experience (QoE), urgency, SLA compliance, and resource utilization. These semantic indicators are used to define the state vector for the agent and to compute a reward function that aligns agent actions with high-level service goals.

Intent Satisfaction and Decision Loop: At each control interval, the agent receives an extended state vector $\mathbf{s}_t = [\text{Throughput}_i, \text{Latency}_i, \text{PacketLoss}_i, \text{SemanticSuccess}_i, \ldots]$ for each slice i. The DQN outputs a resource allocation vector $\mathbf{a}_t$ specifying per-slice allocation ratios. These allocations are enforced in the NS-3 environment via dynamic scheduling, traffic shaping, and configuration of radio and bearer parameters. The resulting slice performance,

including semantic and classical KPIs, is fed back to the agent, closing the loop from intent to enforcement and feedback.

Reward Computation: The agent's reward r_t is calculated as a weighted combination of slice utility, intent satisfaction, SLA compliance, and network efficiency, ensuring that the learning process remains aligned with both user expectations and network operator objectives.

Below, the architecture of the closed-loop IBN-DQN-Semantic Communication is showcased with this figure. High-level intents are mapped to semantic metrics, forming the agent's state vector. Then, the DQN agent outputs allocation decisions, which are enforced in the NS-3 NR network. Slice KPIs and semantic feedback are used for further intent satisfaction and agent learning.

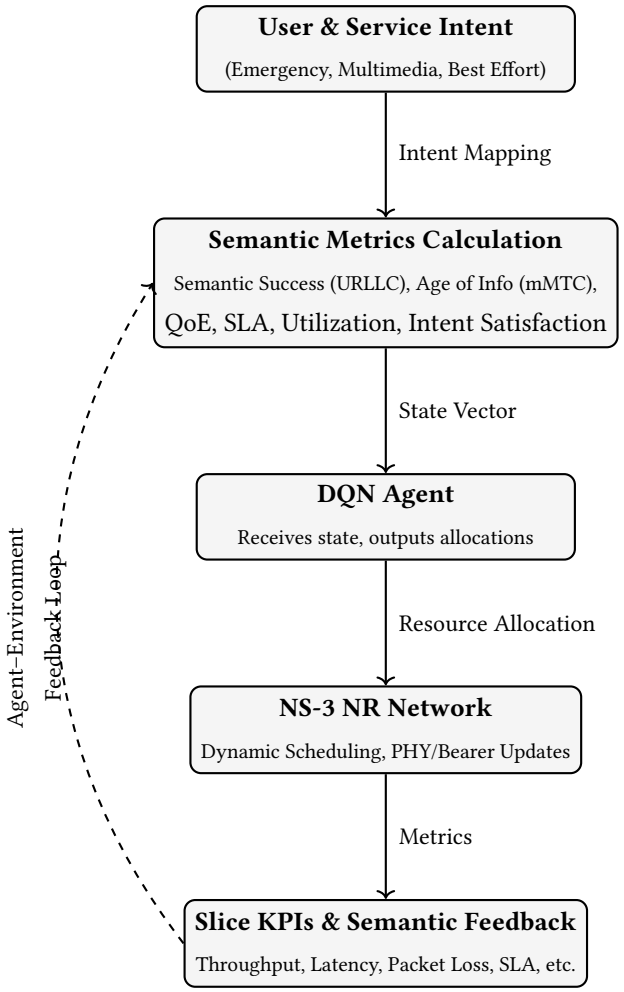


FIGURE 3.2: Closed-loop IBN-DQN-Semantic Communication architecture.

3.4 Semantic Communication Integration

Traditional communication systems have focused on the accurate and efficient transmission of symbols or bits, as established in the Shannon-Weaver model. While this model forms the backbone of modern telecommunications, it does not consider the meaning or usefulness of the transmitted information. Semantic communication introduces the concept of moving beyond syntax to focus on the effectiveness of communication in terms of conveyed meaning, intent and task relevance.

The concept of semantic communication was first formalized by Weaver in the 1940s, who recognized the "semantic problem" of how well the received meaning matches the intended message. Renewed interest in semantic systems, driven by advances in artificial intelligence and edge computing has led to the integration of semantic principles within mission-critical wireless networks such as 5G and beyond. In this context, semantic communication seeks to reduce unnecessary transmissions and optimize for task completion rather than raw throughput.

Within the developed NS-3 simulation framework, a dedicated semantic layer was implemented to enable semantic-aware resource allocation. This layer computes a set of normalized, high-level semantic features for each network slice that captures not only the physical performance (latency, throughput, resource usage) but also contextual meaning, user intent and task fulfillment.

The system supports the three 5G use cases that are represented in the slices: Ultra-Reliable Low-Latency Communications (URLLC), Enhanced Mobile Broadband (eMBB) and massive Machine-Type Communications (mMTC). For each slice, semantic indicators such as Quality of Experience (QoE), intent satisfaction and task completion are computed using context-aware formulas. System-wide metrics are also calculated, including , including network efficiency, urgency and SLA compliance,. All indicators are normalized to the range $[0, 1]$ and updated at each simulation timestep to form a comprehensive semantic state vector passed to the reinforcement learning agent.

On the agent side, these semantic features are incorporated into the state vector and directly used to compute a semantic-aware reward. The reward function is designed to align resource allocation with user intent, service-level agreements and network efficiency, while also considering classical metrics such as throughput and latency. The result is a learning process that is interpretable and directly linked to high-level application goals.

Table 3.2 summarizes all semantic indicators used in the simulation and decision-making process.

Table 3.2: Semantic Indicators and Their Functional Roles

Indicator	**Purpose**	**Range**	**Description**
URLLC Semantic Success	Timely delivery of critical packets	$[0, 1]$	Fraction of URLLC packets delivered within a 1 ms latency constraint
eMBB Semantic Ratio	Contextual quality for video streams	$[0, 1]$	Throughput normalized to a 30 Mbps target for multimedia
mMTC Age of Information	Data freshness for IoT devices	$[0, 1]$	Penalizes stale or outdated IoT measurements
Intent Satisfaction	Dynamic scenario alignment	$[0, 1]$	Composite score that adapts to scenario (e.g., favors URLLC during emergencies)
URLLC QoE	User experience for low-latency apps	$[0, 1]$	Based on URLLC latency under 2 ms
eMBB QoE	Multimedia service quality	$[0, 1]$	Reflects eMBB throughput compared to a 25 Mbps target
mMTC QoE	IoT slice performance score	$[0, 1]$	Based on mMTC latency under 150 ms
Urgency Level	Context-driven task priority	$[0, 1]$	Encodes urgency signal based on network and slice KPIs
URLLC Completion	Mission/task completion under latency	$[0, 1]$	Proportion of flows meeting 1.5 ms deadline
eMBB Completion	Multimedia task completion	$[0, 1]$	Fraction of eMBB flows exceeding 20 Mbps throughput

Continued on next page

Indicator	Purpose	Range	Description
mMTC Completion	IoT update success rate	$[0, 1]$	Proportion of IoT flows with latency under 200 ms
Network Efficiency	Resource use efficiency	$[0, 1]$	Throughput per resource used, measuring efficiency
Congestion Level	Network saturation/load	$[0, 1]$	Ratio of resources in use across all slices
SLA Compliance	Fulfillment of service contracts	$[0, 1]$	Average completion of SLA goals for all slices

All features are computed at each simulation timestep and included in the agent's state vector, enabling real-time and intent-aware control to provide a richer basis for intelligent network management.

Chapter 4

Basic Principles and Implementation Framework for an intelligent DRL-based agent to support IBN capable network slicing with a semantic layer.

4.1 Problem Statement

In fifth generation networks, network slicing is used to support different service types such as URLLC, eMBB and mMTC. However, recent work as shown in the literature review part of this thesis mainly deals with static or pre-configured slice resource allocation, which is insufficient in highly dynamic or emergency scenarios. Fixed slicing cannot accomodate sudden surges in traffic, leading to QoS degradation when conditions change rapidly. The very premise of network slicing is to allocate resources instantly based on current demand, yet networks still lack mechanisms to do this optimally in real time [39]. During a public safety emergency, a flood of URLLC traffic may require immediate resource allocation, something manual and static methods fail to provide. Although research has progressed significantly, current solutions do not fully realize adaptive slicing control under extreme conditions. Machine learning-based studies simplify the problem by evaluating algorithms on static networks operating under ideal conditions. Existing work assumes fixed user locations or constant traffic patterns, achieving good performance only in these constrained settings. These approaches often neglect features of complex networks such as user mobility, time-varying channel quality and advanced 5G New Radio capabilities (e.g. beamforming)

and are rarely taken into consideration together. In other studies, the impact of mobility and interference is examined in isolation but without the integration of an AI agent to manage resources under such conditions. As a result, existing slice optimization algorithms remain largely untested in a truly heterogeneous, time-evolving environment. There is a gap between simulations and real-world complexity and it is uncertain whether a policy trained on a simplistic scenario would maintain performance when faced with the nuanced challenges of a live 5G RAN. A further critical shortcoming in the state of the art is the lack of semantic awareness and intent-driven control. Modern RAN slicing mechanisms handle traffic primarily by separating between service types (e.g. URLLC vs. eMBB) without comprehending the meaning and the importance of data. For instance, an emergency alert video stream and a noncritical video might both be labeled as eMBB, meaning the network would allocate similar resources without realizing one is far more urgent. Likewise, in IBN paradigms where high-level intents may be something similar to "emergency services receive top priority", the RAN slicing enforcement has not been achieved in real time. The combination of semantic meaning and network intent remains an open challenge as current systems do not automatically prioritize traffic based on the actual significance of its content. In an emergency situation, this gap could lead the network to a failure in expediting life-saving communications due to its inability to distinguish them from ordinary traffic under the same slice category. It is evident that a more intelligent approach is needed to identify and prioritize mission critical information in real time, beyond what fixed QoS rules can achieve. To sum up, the probelm addressed by this thesis is how to realize a real-time and intelligent resource allocation scheme for 5G network slicing that can dynamically adapt to emergency conditions and variable traffic., in a way that surpasses current solutions. This entails designing a system able to reallocate radio resources among slices for different situations and incorporate semantic knowledge of the traffic's importance while also following intent-based policies without human intervention. The solution of this problem would enable 5G NR networks to autonomously handle scenarios like disaster response or sudden surges in critical communications, ensuring that high-priority services maintain their required performance while efficiently utilizing network capacity. This thesis proposes a novel solution to achieve these goals by merging deep reinforcement learning with semantic communication and IBN principles in a realistic simulation framework.

4.2 Proposed Solution Architecture

To overcome the challenges discussed, this thesis proposes a novel slicing framework combining Double Deep Q-Network (DQN) reinforcement learning, a semantic communication layer and Intent-Based Networking principles within a high-fidelity simulation environment. The architecture is implemented as a closed-loop system in which a

Python-based (TensorFlow and Keras) agent interacts in real time with an NS3 5G Network simulator. This design directly addresses the limitations of prior works by enabling live, semantic-aware control of slice resources in a realistic RAN setting.

The simulation is a network environment built on the official NR module of NS3 that features a 7 gNodeB hexagonal layout and 12 UEs. This provides a realistic 5G RAN scenario with interference between neighboring cells and spatial diversity. Propagation and radio models include realistic fading, path loss and beamforming to mimic true wireless conditions. Traffic is seperated into different slices that include URLLC, eMBB and Best effort slice. The first two slices are used for emergency voice and video broadcasting using proper NR traffic patterns and the third slice is mostly for generic and IoT traffic. Crucially, an NS-3 Flow Monitor is used to track per-flow performance metrics (latency, throughput, packet loss, SLA compliance etc.) in real time. By capturing these detailed QoS measurements, the simulation closely reflects real-world network behavior and provides the learning agent with feedback on whether slice requirements are being met.

The Deep Reinforcement Learning agent orchestrating resource allocation observes the network state and periodically makes decisions to adjust slicing. The agent is a double Deep Q-Network that uses two neural networks to improve stability by mitigating Q-value overestimation. In practice, the agent maintains an online Q-network which selects actions and a target Q-network that provides stable target values for learning updates. The agent is trained continuously during simulation via rewards that reinforce the desired outcomes such as high reward for meeting URLLC latency targets and penalties for slice SLA violations. Notably, prior research has indicated that DQN-based algorithms are effective for adaptive network slicing in high-dimensional state spaces, which support the choice of this implementation to use a DQN agent. In this solution, the agent's action space corresponds to slice resource allocation decisions (e.g. the agent can choose to momentarily boost URLLC resources at the expense of eMBB or the opposite).

A state of the art feature of the proposed system is the included semantic-aware layer. This module interprets and labels incoming traffic based on its meaning or importance. The data packets are characterized by features presented in table 3.2, in addition to the usual slice classification.These semantic labels are embedded into the state representation given to the DQN agent. By incorporating semantic context, the agent can distinguish truly important traffic from less crucial traffic even if they belong to the same broad QoS category. This enables prioritization of semantically critical flows when resources are constrained. In contrast, existing slice controllers (such as NEMI[17] or 6G-LEGO[34]) do not leverage packet-level meaning for decision making, rather they would treat all URLLC flows equally even if only some carry life-critical information. This approach fills this void, if an emergency URLLC flow is present, the agent is made aware of its urgency and can dynamically allocate more resources to that flow's slice. The semantic layer thus ensures that network decisions are influenced not just by raw QoS requirements, but also by the meaning

of data which is aligned with the emerging vision of semantic networking.

Moreover, the slicing framework presented in this work is designed to be IBN-compatible, meaning it can interpret and fulfill high-level intent policies automatically. The core idea is that the combination of semantic awareness and the learning agent serves as an engine to enforce operator intents in real time. An intent such as "minimize latency for all emergency-related communications" would be interpreted as a change in the reward function to include high penalty for emergency traffic delay and in the semantic tagging of emergency flows to enable the agent to inherently learn to satisfy the intent. The closed-loop control ensures that as soon as an emergency arises, the agent's decisions tilt the network toward that intent by providing more bandwidth or radio resources in general over another slice. Unlike traditional approaches where an operator's intent might require manual reconfiguration or static priority rules, here the adaptation is autonomous and immediate. The agent's behavior aligns with the specified policies without explicit per-event intervention. This tight integration of semantic-driven RL with IBN principles yields what is essentially an intent-driven slicing policy, the network is able to perceive the goal (such as protecting emergency traffic) and the AI controller executes it continuously. This is a significant advance toward self-driving networks that can enforce human intents in a fine-grained and real-time manner.

To combine and implement the discussed concepts in a realistic setting required bridging the C++ NS3 simulator with the Python-Based RL agent. This was achieved through a high-speed shared memory interface that allows the two processes to exchange data (states and actions) at runtime. The mechanism that was integrated is based on the NS3-ai module[1] for NS3, where the simulator exposes certain memory buffers that the agent can read from/write to on each decision loop. Specifically, the agent's observation (state vector), including standard network stats and semantic indicators is written by NS3 into shared memory and the agent's chosen action is returned the same way for NS3 to read and apply. This approach avoids the latency of slower communication methods (such as socket APIs or file I/O) and keeps the simulation and agent tightly synchronized. As a result, the system achieves true real-time interaction where the simulation progresses in step with the agent's decision-making cycle to assist with achieving instant slice reconfigurations at fine time granularity. The shared memory-based co-simulation is a major implementation innovation as most prior research either performs training offline or uses simplified emulator setups. In this framework, the network and the AI agent evolve together in one loop to test how the agent would perform in a live network scenario that includes detailed protocol dynamics. In essence, the architecture enables the simultaneous evolvement of the network and the agent, as the network state changes, the agent adapts in real time and as the agent's policy shifts, so does the network's performance.

[1] https://github.com/hust-diangroup/ns3-ai

The following table highlights the key differences between conventional approaches and the proposed solution across several dimensions.

Table 4.1: Comparison Between Conventional Approaches and the Proposed Solution

Aspect	**Conventional Approaches (State of the Art)**	**Proposed Solution in This Thesis**
Real-time Adaptation	Typically rely on offline-trained models or static slicing configurations, lacking the ability to reallocate resources *during* runtime. Changes in traffic (e.g. emergencies) are handled reactively by human intervention or not at all in real time.	Closed-loop integration of RL with the simulator enables on-the-fly slice adjustment at runtime. The agent continuously adjusts resources each scheduling interval via shared memory feedback, achieving genuine real-time control.
Network Environment	Often evaluated in simplified or static settings – e.g. single-cell or small network models with fixed users. Mobility, interference, and advanced 5G features are frequently ignored, limiting realism.	Uses a realistic 5G NR environment with a 7-cell hexagonal layout, moving UEs, and full PHY dynamics (path loss, fading, handovers). Key NR features like beamforming and power control are included to emulate real-world behavior. This high-fidelity environment ensures the agent is tested under real network conditions.

Aspect	Conventional Approaches (State of the Art)	Proposed Solution in This Thesis
Semantic Awareness	No utilization of semantic information about traffic; slices treat all data within a QoS class uniformly. For example, an emergency message and a normal packet in the URLLC slice are handled the same, since conventional policies lack content awareness.	Introduces a semantic layer that classifies flows by meaning/priority (e.g. critical alert vs. routine data). These semantic tags become part of the agent's input, so the slicing decisions explicitly prioritize *semantically important* traffic during congestion. This ensures truly critical communications get preferential treatment beyond what static QoS labels provide.
Intent-Based Control	High-level operator intents (e.g. "guarantee latency $<$ X for service Y") are not directly translated in real time. Network policies remain largely static or require manual reconfiguration; there is little integration of intent frameworks with slice controllers in current practice.	Aligns with Intent-Based Networking by dynamically enforcing intents through the agent's reward and decision logic. The system automatically maps intents (such as prioritizing emergency services) into slice resource actions. This creates an autonomous, closed-loop enforcement of intents – the agent continuously tunes the network to meet specified SLA goals without human intervention.

Aspect	Conventional Approaches (State of the Art)	Proposed Solution in This Thesis
Learning Algorithm	Some prior works apply DRL to slicing, but often using standard DQN or policy-gradient methods without enhancements, which can struggle in complex, non-stationary environments. Inadequate handling of exploration or convergence issues can lead to unstable or suboptimal policies.	Utilizes a **Double DQN** algorithm to improve learning stability (using separate target networks to reduce value overestimation). Also employs **prioritized experience replay** so that important events (like emergency spikes) are learned from more frequently. These enhancements make the agent more robust and efficient in learning the optimal slicing policy under dynamic conditions.
Integration & Implementation	Many solutions are evaluated in isolated simulations or theoretical models. There is usually no live interface between the learning agent and a network simulator – either the environment is abstracted, or training data is generated offline and fed into an ML model later. This gap raises questions about real-world deployability.	Fully integrates a live NS-3 simulation with a Python RL agent via a shared memory interface. The agent and network simulator run in sync, exchanging state and actions at runtime. This high-fidelity co-simulation demonstrates that an AI agent can directly control a network in real time, providing a realistic path toward deployment in actual network controllers.

Aspect	Conventional Approaches (State of the Art)	Proposed Solution in This Thesis
Performance Metrics	Evaluations often focus on aggregate metrics like throughput or utility. Fine-grained QoS outcomes per slice (latency, jitter, SLA violation rate) may not be thoroughly examined, especially if the environment is simplistic.	Tracks detailed **flow-level and slice-level metrics** throughout the simulation – latency distribution, packet delivery ratio for URLLC, throughput for eMBB, etc. – using NS-3's built-in monitoring. This comprehensive measurement confirms whether the agent truly meets the intent (e.g., URLLC latency bounds) and quantifies improvements over static slicing.

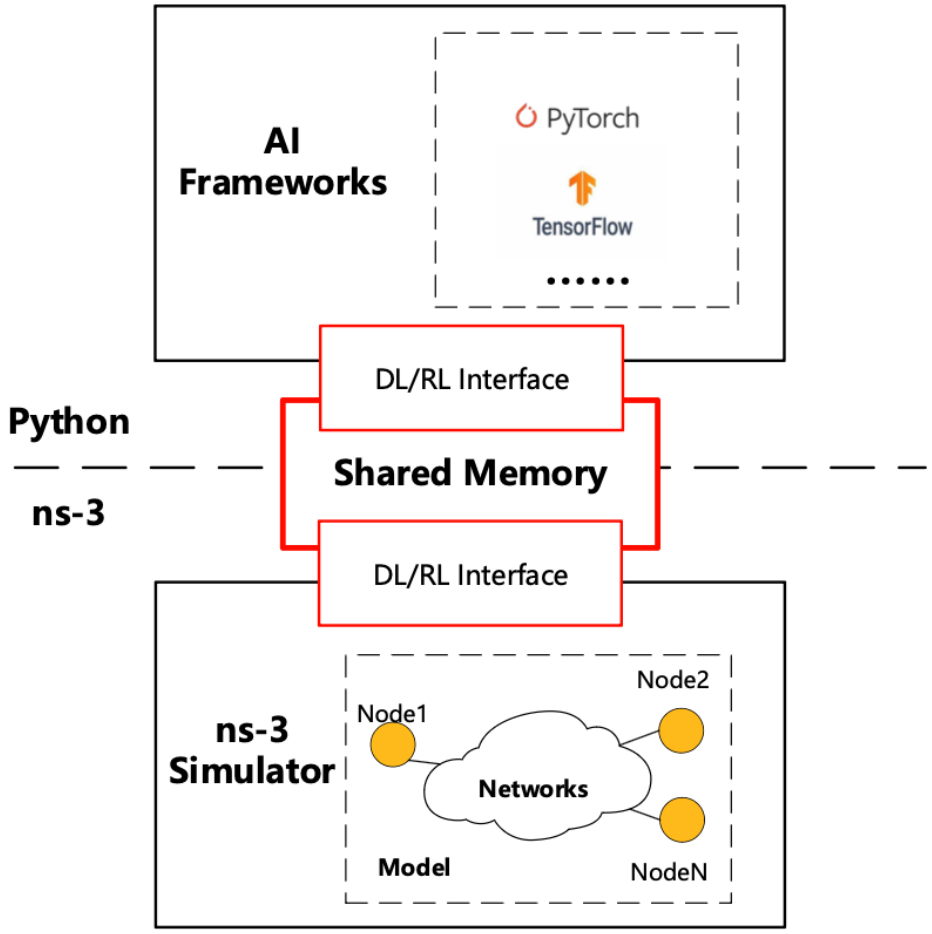


FIGURE 4.1: Official representation of the Shared Memory interface of NS3-ai.

4.3 System Workflow and Implementation

In this section, the workflow of the proposed system is presented in a figure. The architecture follows a closed-loop design in which the high-level intents of operators are simulated in the NS3 NR environment and processed by the reinforcement learning agent for real-time adaptation. During the process Key Performance Indicators (KPIs) and semantic metrics are continuously fed to the agent, enabling dynamic policy creation that aligns network behavior with specified service objectives.

The following figure presents the complete workflow for the proposed system. User/operator intent is mapped to semantic priorities and flows (left). NS-3 simulates 5G traffic, extracts KPIs and semantic metrics, exports state rightwards to the RL agent (right column) via shared memory.The Python Double DQN agent outputs resource allocations with KPIs and semantic feedback closing the loop for adaptation.

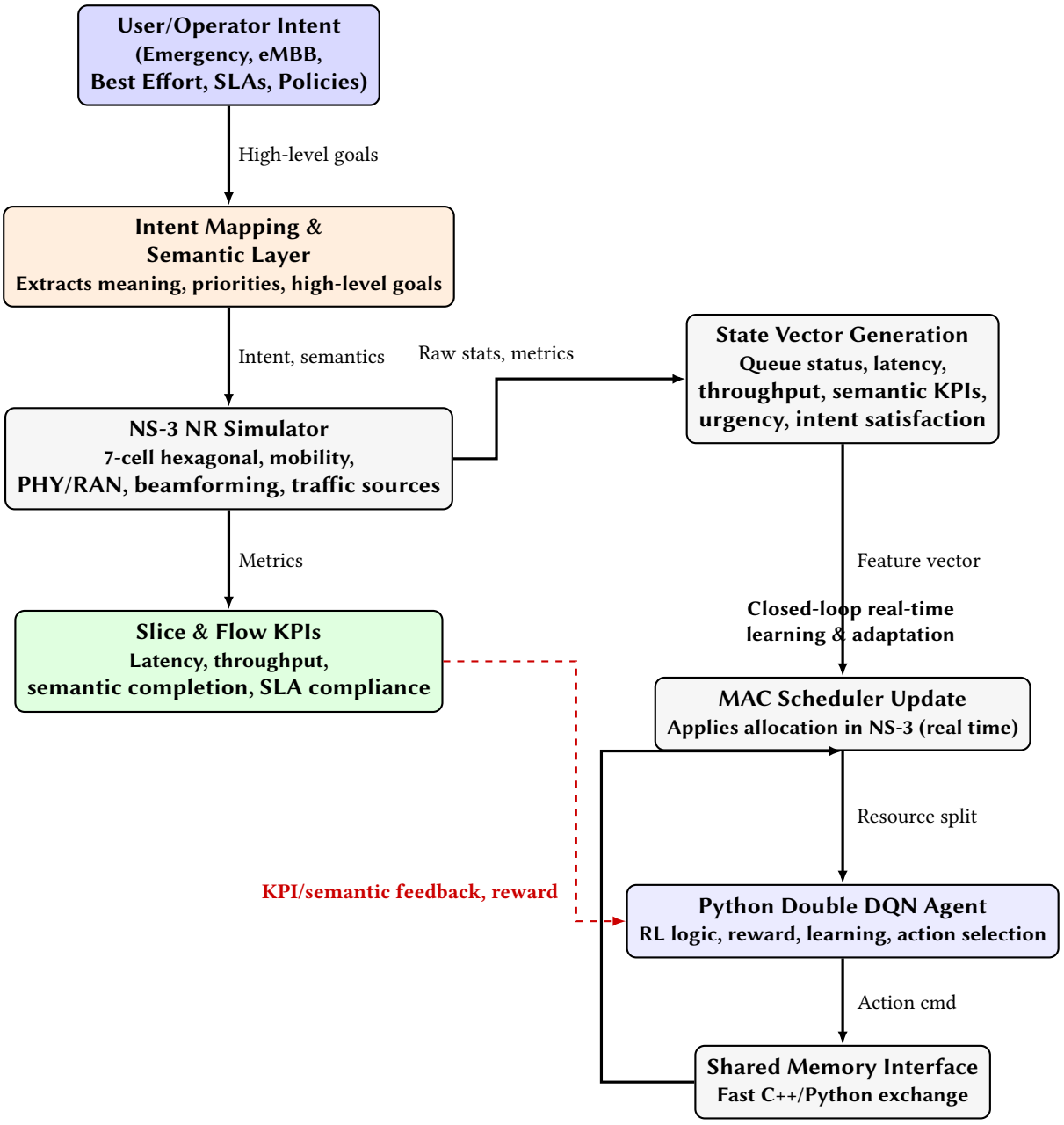


FIGURE 4.2: Complete end-to-end workflow for the system.

Chapter 5

Implementation and Core Components of the solution

5.1 Agent Interface, State & Action Definition, and Reward Shaping (5.1)

This section specifies the exact Python–C++ interface and learning signals used by the agent in the system. The ns-3 side exports a packed `SlicingState` struct via the `ns3-ai` shared-memory bridge, which is consumed by the Python agent (`TrueDQNAgent`). The agent converts it to a normalized numerical state vector, selects a slicing action and then writes back a `SlicingAction` struct with slice resource shares for URLLC, eMBB and mMTC.

Control interval. The closed-loop runs once per simulator control step (the main loop in `agent.py`). In the experiments carried out, an agent's decision corresponds to approximately $\Delta t \approx 0.5\,\mathrm{s}$ of simulated time, that is, one control interval per step.

State vector. At step t, the agent receives

$$\mathbf{s}_t \in \mathbb{R}^{21}, \qquad \mathbf{s}_t = \big[\ \underbrace{\mathbf{c}_t}_{\text{core (7)}}, \ \underbrace{\mathbf{z}_t}_{\text{semantic (14)}}\ \big],$$

where $\mathbf{c}_t$ are core RAN KPIs and $\mathbf{z}_t$ are semantic/intent features. The exact order and meaning match the exported fields of `SlicingState` and the `get_state_vector` in the agent. The semantic indicators are computed in C++ and normalized to $[0, 1]$ before being handed to Python.

TABLE 5.1: State vector features exported in `SlicingState` and consumed by the agent (order is exact).

Field (code name)	**Description**	**Units / Range**
`throughputUrllc`	Slice throughput for URLLC flows (aggregated)	Mbps
`throughputEmbb`	Slice throughput for eMBB flows (aggregated)	Mbps
`throughputMmtc`	Slice throughput for mMTC flows (aggregated)	Mbps
`latencyUrllc`	Mean one-way latency for URLLC	ms
`latencyEmbb`	Mean one-way latency for eMBB	ms
`latencyMmtc`	Mean one-way latency for mMTC	ms
`resourceUtilization`	Current gNB resource usage level	$[0, 1]$
`urllcSemanticSuccess`	Timely delivery ratio for URLLC critical packets (semantic success)	$[0, 1]$
`embbSemanticRatio`	Contextual quality ratio for eMBB (e.g., vs. a 30 Mbps video target)	$[0, 1]$
`mmtcAgeOfInfo`	Freshness score for IoT updates (higher is fresher)	$[0, 1]$
`intentSatisfaction`	Alignment with current operator/user intent	$[0, 1]$
`urllcQoE`	QoE score for URLLC slice	$[0, 1]$
`embbQoE`	QoE score for eMBB slice	$[0, 1]$
`mmtcQoE`	QoE score for mMTC slice	$[0, 1]$
`urgencyLevel`	Urgency indicator for emergency context	$[0, 1]$
`urllcCompletion`	Fraction of URLLC flows meeting deadline	$[0, 1]$
`embbCompletion`	Fraction of eMBB flows hitting minimum rate	$[0, 1]$
`mmtcCompletion`	Fraction of mMTC flows meeting latency bound	$[0, 1]$
`networkEfficiency`	Throughput-per-resource efficiency metric	$[0, 1]$
`congestionLevel`	System congestion level	$[0, 1]$
`slaCompliance`	Aggregate SLA satisfaction across slices	$[0, 1]$

The Python agent uses the above *as-is* to preserve the order and ranges produced by the C++ side.

Action space and scheduler knobs. The agent outputs a *slice share* vector

$$\mathbf{a}_t = \left[a_t^{\mathrm{URLLC}},\ a_t^{\mathrm{eMBB}},\ a_t^{\mathrm{mMTC}}\right] \in \Delta^2, \quad a_t^{(i)} \in [0,1],\ \sum_i a_t^{(i)} = 1,$$

which is written into `SlicingAction{allocUrllc, allocEmbb, allocMmtc}`. These shares are interpreted by the ns-3 scheduler as per-slice weights over available RBs. The agent's discrete action set spans allocations in 0.1 increments (simplex grid), so the effective per-step change is implicitly bounded by this discretization.[1]

Intent targets and safety guardrails. We denote the eMBB informational throughput target as $T_{\mathrm{inf}} = 30\,\mathrm{Mbps}$ (NGMN-class video). To guarantee mission-critical performance under emergencies, we use:

- **Minimum URLLC share rule:** a fallback allocation of $[0.4,\ 0.4,\ 0.2]$ is enforced if an invalid action is produced; in adaptive mode, the agent targets

  $$a_t^{\mathrm{URLLC}} \in \{0.6,\ 0.7,\ 0.8\}$$

  based on a severity score $\sigma_t = \min(\texttt{latencyUrllc}/10,\ 3)$ (low/med/high).

- **Action validity:** actions are renormalized to the probability simplex and clamped to $[0, 1]$ if needed.

Reward shaping. The scalar reward r_t computed in `calculate_reward` integrates hard URLLC latency constraints, smoothed eMBB throughput shaping, mMTC throughput thresholds, allocation-based bonuses, diversity and anti-repetition incentives, improvement/stagnation tracking, and semantic/intent-aware components, plus mild regularization. The URLLC term is a 7-level piecewise function of $\ell = \texttt{latencyUrllc}$ (ms), where a stricter penalty is applied in the 5–10 ms band to discourage operation in this

[1] There is no additional hard delta cap; however, action diversity and anti-stagnation terms in the reward discourage abrupt oscillations.

"safety margin":

$$R_{\mathrm{URLLC}} = \begin{cases} -1000, & \ell > 50, \\ -500, & 20 < \ell \le 50, \\ -200, & 10 < \ell \le 20, \\ -500, & 5 < \ell \le 10, \\ +50, & 2 < \ell \le 5, \\ +100, & 1 < \ell \le 2, \\ +200, & \ell \le 1. \end{cases}$$

The eMBB reward uses a smoothed logistic centered at $400\,\mathrm{Mbps}$:

$$\tilde{T}_{\mathrm{e}} = \begin{cases} 4\,T_{\mathrm{e}}, & T_{\mathrm{e}} < 10, \\ T_{\mathrm{e}}, & \text{otherwise}, \end{cases} \qquad R_{\mathrm{eMBB}} = 400 \cdot \frac{1}{1 + \exp\left[-0.03\,(\tilde{T}_{\mathrm{e}} - 400)\right]} - 200 + R^{\mathrm{e}}_{\mathrm{sustain}} + R^{\mathrm{e}}_{\mathrm{instab}},$$

where $T_{\mathrm{e}} = \texttt{throughputEmbb}$. The sustained bonus $R^{\mathrm{e}}_{\mathrm{sustain}} = 50 \cdot s$ if $\tilde{T}_{\mathrm{e}} \ge 350$ for $s > 2$ consecutive steps, and the instability penalty $R^{\mathrm{e}}_{\mathrm{instab}} = -100$ if $\tilde{T}_{\mathrm{e}}$ drops by more than $50\,\mathrm{Mbps}$ compared to the previous step.

The mMTC component is stepwise:

$$R_{\mathrm{mMTC}} = \begin{cases} +50, & T_{\mathrm{m}} > 0.1, \\ +20, & T_{\mathrm{m}} > 0.01, \\ -20, & \text{otherwise}, \end{cases}$$

with $T_{\mathrm{m}} = \texttt{throughputMmtc}$ (Mbps).

Smart allocation bonuses R_{alloc} are awarded if the chosen allocation proportions $a^{\mathrm{URLLC}}, a^{\mathrm{eMBB}}$ exceed slice-specific thresholds when the slice is underperforming: for example, if $\ell > 30\,\mathrm{ms}$ and $a^{\mathrm{URLLC}} > 0.6$ then $+300$; if $\tilde{T}_{\mathrm{e}} < 50\,\mathrm{Mbps}$ and $a^{\mathrm{eMBB}} > 0.6$ then $+300$, with smaller bonuses for milder conditions.

Diversity and anti-repetition bonuses R_{div} are computed from the rounded allocation tuple: first occurrence $+100$, then $+50$, $+20$, 0, and after five repeats -50. Patterns over the last three actions are penalized if repetitive (AAA $\to -100$, ABA $\to -50$).

Improvement/stagnation tracking R_{imp} adds $+100$ or $+50$ for significant latency or throughput improvements, subtracts -50 for large degradations, and applies a -50 stagnation penalty if changes are negligible and the action is repeated.

Semantic-aware terms include: Semantic-aware terms include:

$$
\begin{aligned}
R_{\text{intent}} &= 150 \cdot \texttt{intentSatisfaction}, \\
R_{\text{QoE}} &= 50 \cdot (\texttt{urllcQoE} + \texttt{embbQoE} + \texttt{mmtcQoE}), \\
R_{\text{semantic}} &= 100 \cdot \texttt{urllcSemanticSuccess} + 80 \cdot \texttt{embbSemanticRatio} \\
&\quad + 60 \cdot \texttt{mmtcAgeOfInfo}, \\
R_{\text{SLA}} &= 100 \cdot \texttt{slaCompliance}, \\
R_{\text{eff}} &= 80 \cdot \texttt{networkEfficiency} - 50 \cdot \texttt{congestionLevel}.
\end{aligned}
$$

The final reward is

$$
\begin{aligned}
r_t = \kappa \,\mathrm{clip}\Big(& R_{\text{URLLC}} + R_{\text{eMBB}} + R_{\text{mMTC}} + R_{\text{alloc}} + R_{\text{div}} + R_{\text{imp}} \\
& + R_{\text{intent}} + R_{\text{QoE}} + R_{\text{semantic}} + R_{\text{SLA}} + R_{\text{eff}}, \\
& -2000,\ 2000 \Big), \quad (5.1)
\end{aligned}
$$

where κ is a global scaling factor (`reward_scale`). This shaping intentionally focuses on URLLC latency safety while still rewarding eMBB informativeness and IoT and incorporates semantic KPIs for intent fulfillment and SLA compliance.

IBN/semantics coupling. High-level intent is ingested as semantic scores on the C++ side (e.g., `intentSatisfaction`, `urgencyLevel`) and directly participates in r_t, so the agent's policy aligns with operator goals without manual reconfiguration. Likewise, semantic fields (`urllcSemanticSuccess`, `embbSemanticRatio`, `mmtcAgeOfInfo`) bias decisions toward meaning-aware resource use rather than raw KPI maximization.

Summary. Concretely, the environment-to-agent interface is: (i) a 21D state vector with the exact fields and ranges listed in Table 5.1; (ii) a 3D action on the probability simplex mapping to scheduler weights; (iii) a shaped reward that enforces URLLC latency, drives eMBB toward informative throughput ($T_{\text{inf}} = 30$ Mbps) and honors intent priorities. Safety guardrails guarantee robust behavior under emergencies and unstable conditions, all executed at a control interval of $\sim$0.5 s per step.

5.2 Neural Network Architecture (5.2)

Design goal. The agent must fuse low-level radio KPIs with high-level intent signals and output discrete slice shares. We therefore adopt a light *dueling Double-DQN* with prioritized replay.

Input representation. The state is a 21-dimensional vector $\mathbf{s} \in \mathbb{R}^{21}$ composed of 7 core KPIs and 14 semantic features (see §5.1, Table 5.1). Values are pre-normalized on the C++ side and consumed *as-is* by the Python agent.

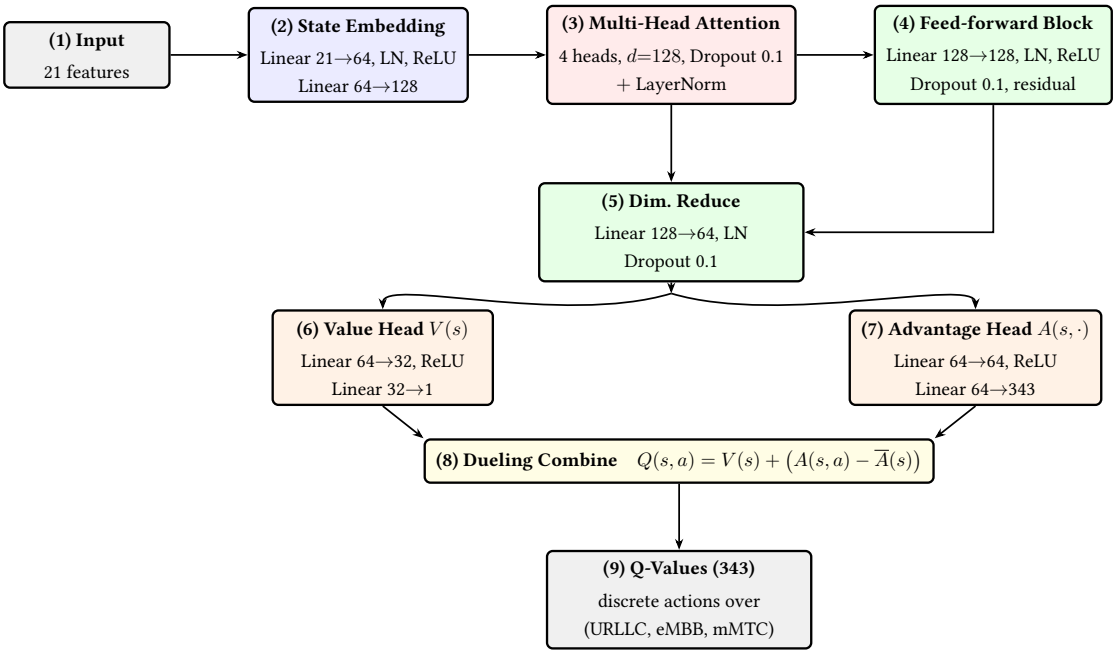


FIGURE 5.1: Block diagram of the attention-augmented, dueling DQN.

(1–5) Encoder. A shallow *state embedding* maps $\mathbf{s}$ to a 128-dimensional representation using linear layers, layer normalization (LN) and ReLU. A lightweight *multi-head self-attention* refines the contextual mix between KPIs and semantics, followed by a residual feed-forward block and a dimensionality reduction to 64.

(6–9) Dueling head and outputs. From the 64-dimensional bottleneck, two heads are computed:

$$V(s) \in \mathbb{R}, \qquad A(s,\cdot) \in \mathbb{R}^{343},$$

and combined as

$$Q(s,a) = V(s) + \big(A(s,a) - \overline{A}(s)\big), \quad \overline{A}(s) = \tfrac{1}{343}\sum_{a'} A(s,a').$$

Each of the 343 outputs corresponds to a discrete allocation triple (URLLC, eMBB, mMTC) on a $7 \times 7 \times 7$ simplex grid (see §5.1 for the action encoding and guardrails).

Training and stability. We employ Double-DQN with a target network θ^- and soft updates

$$\theta^- \leftarrow \tau\,\theta + (1-\tau)\,\theta^- \quad \text{(every 3 training steps, } \tau = 0.15\text{)}.$$

Transitions are sampled from a prioritized replay buffer of capacity 20,000 with exponent $\alpha = 0.6$ and importance sampling $\beta = 0.4$. Optimization uses AdamW (learning rate 10^{-3}, weight decay 10^{-4}) with cosine annealing; dropout 0.1 and layer normalization are applied throughout. Discount $\gamma = 0.95$. Exploration follows an ε-greedy policy with $\varepsilon_0 = 0.7$, decay 0.98 per step, floor 0.1. Mini-batches of size 4 are used for on-policy cadence with the simulator.

Effectiveness of layout. The encoder (1–5) mixes KPIs and semantics using attention while keeping compute light; the 64-dimensional bottleneck limits overfitting. The dueling head (6–9) stabilizes value estimation in a large discrete action space, which is essential for real-time slicing where many allocations are near-equivalent. Prioritized replay amplifies learning on rare but critical events (e.g., URLLC emergencies) captured by the semantic features.

5.3 DQN Training Methodology in NS-3

The Deep Q-Network (DQN) agent was trained within the NS-3 simulation environment using a continuous single-episode framework, where the episode length corresponded to the total duration of the simulated scenario. Each control decision was taken at discrete intervals of 0.5 s of simulated time that are referred as *steps*. At every step, the agent received a complete representation of the network state from the NS-3 side through the `msgInterface.GetCpp2PyStruct()` mechanism, which facilitated the transfer of a structured C++ data object to Python.

Upon receiving the current state, the agent selected an action according to its current policy and then communicated it back to NS-3 for execution. The environment returned the next state and the corresponding reward, after which the agent stored the resulting experience tuple $(s_t, a_t, r_t, s_{t+1}, d_t)$ in a prioritized replay buffer. The replay buffer implementation supported prioritized sampling, with sampling probabilities adjusted based on the temporal-difference (TD) error of each stored transition. This ensured that more informative experiences were replayed more frequently, improving sample efficiency and training stability. After each step, the `replay()` method was invoked to sample a batch of

experiences to perform a gradient update on the Q-network. Updated TD errors were written back to the buffer to maintain the prioritization scheme.

Moreover, the state vector provided to the agent incorporated both conventional key performance indicators (KPIs) and semantic communication metrics. Seven KPIs were included, covering slice-specific throughput and latency for URLLC, eMBB and mMTC, along with overall resource utilization. Fourteen additional semantic features were computed directly in the C++ simulation, encompassing measures such as intent satisfaction, Quality of Experience (QoE) for each slice, urgency levels, completion rates, network efficiency, congestion and Service Level Agreement (SLA) compliance. These features were concatenated into a fixed-length 21-dimensional vector, which served as the input to the neural network for both action selection and training.

Action selection followed an ϵ-greedy exploration policy. With probability ϵ, the agent selected an exploratory action, whereas with probability $1 - \epsilon$ the action corresponding to the highest predicted Q-value was chosen. The exploration rate ϵ multiplicatively decayed after each step according to

$$\epsilon_t = \max\big(\epsilon_{\text{min}}, \epsilon_0 \cdot (\epsilon_{\text{decay}})^t\big),$$

with $\epsilon_0 = 0.7$, $\epsilon_{\text{min}} = 0.1$, and $\epsilon_{\text{decay}} = 0.98$. To prevent convergence to sub-optimal deterministic policies, forced exploration was employed when repeated action patterns were detected, temporarily increasing ϵ to 0.9.

To enhance training stability, a Double DQN framework was adopted, maintaining both a primary Q-network and a separate target network. Target network updates were performed using a combination of hard and soft update strategies: every $N_{\text{update}} = 3$ steps, the target network was updated by blending its parameters with those of the primary network using a polyak averaging factor $\tau = 0.15$.Furthermore, Optimization was carried out using the AdamW optimizer with a learning rate of 10^{-3}, weight decay of 10^{-4}, and cosine annealing learning rate scheduling. This combination was selected to encourage stable convergence while mitigating overfitting.

Convergence was assessed by monitoring both the loss function and the cumulative rewards over time. Additionally, semantic and network performance metrics were periodically evaluated to confirm that policy improvements translated into tangible service-level gains. The agent logged loss values, reward trajectories and key performance indicators to persistent storage for subsequent analysis, enabling verification of learning progress and final policy quality.

5.4 Agent Interaction with the NS-3 Slicing Simulation

The connection between the semantic-aware DQN agent and the NS-3 slicing environment is implemented via a custom Python–C++ integration layer built with `pybind11`. This layer exposes the NS-3 state and action data structures directly to Python while maintaining simulation synchronisation through a shared memory transport channel.

Interface and Data Structures

The C++ module `nr_slicing_py.cc` registers two core structures:

- **`SlicingState`**: Encapsulates 21 core KPIs and semantic metrics.
- **`SlicingAction`**: Stores three allocation variables (`allocUrllc`, `allocEmbb`, `allocMmtc`), representing the fraction of total radio resources assigned to each slice.

Both structures are exposed to Python using `pybind11::class_`, enabling field-level read/write access without serialization overhead.

The message interface is provided by the class `Ns3AiMsgInterfaceImpl<SlicingState, SlicingAction>`, which implements:

- `GetCpp2PyStruct()` – returns a reference to the state struct for Python to read.
- `GetPy2CppStruct()` – returns a reference to the action struct for Python to update.
- `PySendBegin()` / `PySendEnd()` – writes the Python-modified action to shared memory for C++ to consume.
- `PyRecvBegin()` / `PyRecvEnd()` – retrieves the latest state from C++ into Python.

Control Flow and Synchronisation

The NS-3 component `IBNSlicingManager` (`nr-slicing-env.cc`) orchestrates the control loop:

1. At each control interval (≈ 0.5 seconds of simulated time), NS-3 populates a `SlicingState` object with the current KPIs and semantic indicators.
2. The `SendState()` method writes this struct into the shared memory segment using `CppSendBegin()` / `CppSendEnd()`.

3. The simulation halts until an updated `SlicingAction` is received from Python via `ReceiveAction()`, which calls `CppRecvBegin()`, reads the action and resumes execution.

This synchronous exchange ensures that each simulation time step is paired with exactly one agent decision to preserve causal consistency for training and evaluation.

Python Integration Layer

On the Python side, the `Experiment` class in `ns3ai_utils.py` is responsible for:

- Initialising the message interface with user-defined shared memory size (`shmSize`, default 4096 bytes), segment name and message channel identifiers.
- Launching the NS-3 simulation as a subprocess with the specified target script (e.g., `nr-slicing` executable).
- Providing access to `msgInterface` for reading `SlicingState` and writing `SlicingAction`.
- Handling cleanup by terminating the NS-3 process and releasing shared memory on exit.

The shared memory is created by Python (`SetIsMemoryCreator(true)`) and used in non-vector mode (`SetUseVector(false)`), as the data is exchanged as fixed-size structs rather than arrays.

Action Encoding and Decoding

To ensure valid allocation decisions, the agent's neural network outputs a discrete action index in the range $0 \leq a < 343$. This index is decoded into $[\texttt{allocUrllc}, \texttt{allocEmbb}, \texttt{allocMmtc}]$ that sums to 1.0, representing the proportion of radio resources assigned to each slice. These values are written directly to the fields of `SlicingAction` before being transmitted to NS-3.

Technological Rationale

The chosen architecture avoids TCP/UDP communication overhead by using direct shared memory transfer. This solution enables sub-millisecond round-trip latency between NS-3 and the Python agent. The synchronous execution model ensures that training data is temporally

aligned with network events, a requirement for reliable convergence of reinforcement learning algorithms in time-sensitive network control scenarios. Furthermore, the struct-based interface ensures type safety and eliminates costly serialisation/deserialisation steps.

Chapter 6

Experimentation & Validation

6.1 Experiment Setup

The experimental evaluation was conducted using the `ns-3.44` network simulator, augmented with the `NS3-AI` module and the official 5G NR module. The custom slicing solution was implemented within the `contrib/ai/examples/nr-slicing` directory. The host system ran `Ubuntu 24.04 LTS` and `Python 3.12` with `PyTorch 2.1` for the Reinforcement Learning agent. Simulations were executed on a machine equipped with an `Intel Core i7-7700` CPU (8 cores, 3.6 GHz) and 32 GB RAM. No GPU acceleration was used during training.

Network Topology and Radio Configuration

The simulated deployment consisted of 7 gNBs arranged in a single-ring hexagonal layout, serving 12 UEs distributed across the network. The inter-site distance (ISD) was set to 500 m, with gNBs mounted at a height of 25 m and UEs positioned at 1.5 m. The carrier operated at 3.5 GHz with a 100 MHz total bandwidth, divided into three Bandwidth Parts (BWPs), one per slice. Numerology was configured with a 30 kHz subcarrier spacing ($\mu = 2$) and a TDD 4:1 DL:UL ratio. Each gNB was equipped with a 4×4 MIMO antenna array, while UEs had 2×2 MIMO capability. Beamforming followed the 3GPP codebook-based approach included in the NR module. Propagation was modeled using the 3GPP Urban Microcell (UMi) channel model with log-normal shadowing ($\sigma = 4$ dB). UEs moved according to a RandomWaypoint mobility model at a constant speed of 3 km/h.

Traffic-to-Slice Mapping

Three network slices were configured:

- **URLLC**: carrying CBR voice traffic at 100 kbps with 32-byte packets, initiated at 0.1 s and maintained until simulation end.
- **eMBB**: serving OnOff video/FTP traffic at 10 Mbps with 1500-byte packets and a 0.8 duty cycle, starting at 0.1 s.
- **mMTC**: supporting CBR IoT telemetry at 1 kbps with 64-byte packets, starting at 0.1 s.

Applications were explicitly mapped to slices during initialization. URLLC carried voice traffic, eMBB served high-throughput applications, and mMTC supported low-rate telemetry. This mapping was enforced in the NR MAC scheduler via EPS bearer configurations:

- URLLC: QCI 1, ARP 1, preemption enabled
- eMBB: QCI 7, ARP 5
- mMTC: QCI 9, ARP 9

This ensured strict QoS alignment between application type and slice configuration.

The first figure from the simulation results shows the initial per-slice throughput from the start of the simulation. The plot confirms that the traffic-to-slice mapping is active and each slice begins with its expected traffic profile before the agent adapts allocations.

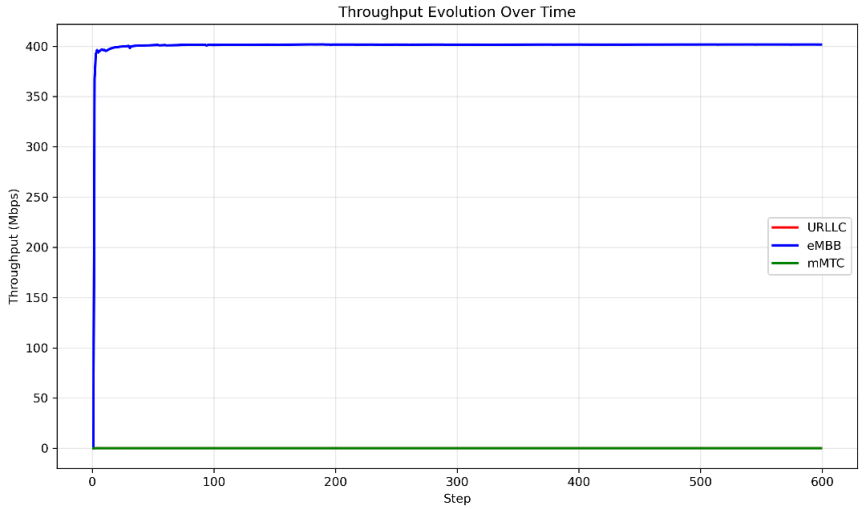


FIGURE 6.1: Initial per-slice throughput.

Control and Training Parameters

The Reinforcement Learning agent operated with a control interval of 0.1 s (allocation decisions updated every 100 ms). Each episode lasts for a predetermined number of seconds in the code, with the first 2 s serving as a warm-up phase without agent intervention. Experiments were repeated multiple times with different parameters.

Baselines and Ablation Studies

To contextualize the agent's performance, two baseline schedulers were evaluated:

1. **Static allocation**: fixed equal shares (33% per slice).
2. **Heuristic allocation**: proportional to observed demand.

One ablation variant of the agent was also tested:

- **No-semantics**: state vector excluded semantic layer metrics.

Metrics and Data Collection

Performance data were obtained from NS-3 FlowMonitor (XML output) and relayed to the Python side. The C++ environment emits:

- `/results/nr_slicing_flowmon.xml` – per-flow statistics.

The explicit mapping of traffic types to slices, combined with deterministic initialization parameters, ensures the experiment can be replicated without ambiguity.

6.2 Data Sources and Measurement Methodology

In this study, no pre-collected or static dataset was employed for either training or evaluation. Instead, all measurements and performance indicators were generated dynamically through real-time interactions between the Deep Q-Network (DQN) agent and the live `ns-3.44` simulation environment. The network model consisted of the official NR module augmented with the `NS3-AI` framework, allowing bidirectional communication between the simulator and the Python-based agent via `PyBind11` bindings, as shown in the LTE-CQI example. At every control interval ($\Delta t = 0.1\,\mathrm{s}$), the agent received a complete

state vector containing physical KPIs, semantic indicators and slice-level statistics and returned an allocation decision. The duration of each episode different for each experiment, for instance during the development phase the episodes were short in duration to test basic functionalities and during validation it was increased to acquire more results.

Collected Metrics and Their Sources

Metrics were extracted from two primary sources:

1. **NS-3 FlowMonitor** — Provided per-slice and per-flow physical KPIs, including:
    - Throughput (Mbps)
    - Latency (ms)
    - Packet loss (%)
2. **Custom Agent Logs** — Generated by the Python agent and the C++ simulation code, capturing:
    - Intent satisfaction (0–1 normalized)
    - Slice-specific Quality of Experience (QoE) for URLLC, eMBB, mMTC (0–1 normalized)
    - SLA compliance (%)
    - Network efficiency (%)
    - Semantic indicators:
        - URLLC semantic success
        - eMBB semantic ratio
        - mMTC age of information
        - Urgency level
        - Completion ratios for all three slices
        - Congestion level
    - Agent-side statistics:
        - Action history (allocation vectors per step)
        - Reward history (per step and per episode)
        - Training loss (per episode)
        - Replay memory buffer utilization
        - Exploration rate (ϵ)

Unit Conventions

Throughput was reported in Mbps, latency in milliseconds, and packet loss as a percentage. Semantic indicators and QoE scores were normalized to the range $[0, 1]$. SLA compliance and network efficiency were expressed as percentages.

Post-Processing and Statistical Treatment

For each configuration, metrics were averaged over all seeds and runs. Semantic indicators and QoE values were normalized prior to aggregation, while physical KPIs retained their original units. Ninety-five % confidence intervals were computed using the standard error across runs. Efficiency and SLA compliance were reported as percentages. All performance curves (throughput, latency, intent satisfaction, SLA compliance) were plotted as mean $\pm$ confidence interval and allocation distributions were visualized as histograms over the action space.

6.3 Performance Metrics & Analysis

Overview The performance of the proposed semantic-aware Deep Reinforcement Learning (DRL) slicing framework was evaluated in terms of both traditional network key performance indicators (KPIs) and semantic-layer metrics. The analysis presented in this section is based exclusively on the simulation run used for validation, which corresponds to a long episode duration. The evaluation combines throughput, latency and allocation behaviour with higher-level semantic metrics such as intent satisfaction, quality of experience (QoE) and service-level agreement (SLA) compliance.

In the following figure, the distribution of semantic metrics is presented including intent satisfaction, QoE, SLA compliance and efficiency across the evaluated episode. The mass concentrated near 1.0 indicates consistently high semantic performance throughout the simulation.

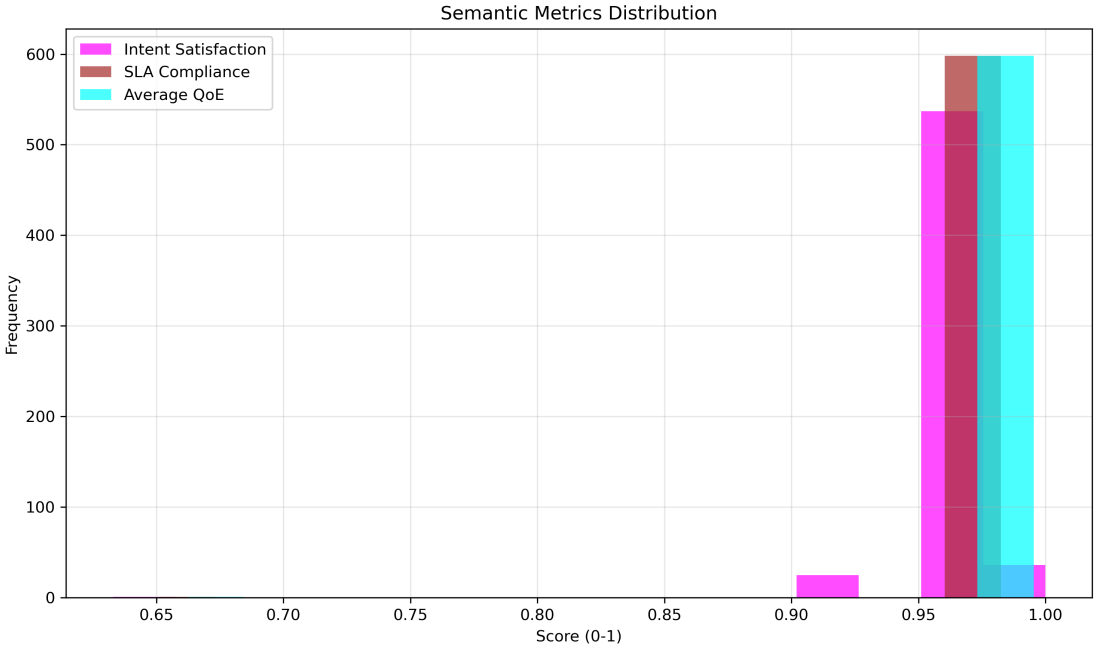


FIGURE 6.2: Distribution of semantic-layer metrics.

Throughput Analysis The throughput evolution indicates that the URLLC and eMBB slices rapidly reached stable performance levels within the first ten seconds of simulation time. The URLLC slice maintained a mean throughput of approximately 3.7 Mbps, closely matching its provisioned capacity for voice-grade low-latency services. The eMBB slice stabilised near its configured allocation for high-throughput applications. The mMTC slice exhibited a stable but low throughput of approximately 0.013 Mbps, in line with its sparse telemetry workload. This stability across slices demonstrates that the agent consistently aligned allocation with slice-specific QoS profiles.

The below figure represents the aggregate boxplot of key KPIs across slices. Narrow variance for URLLC and eMBB throughput and low dispersion overall indicate stable slice behaviour consistent with target profiles.

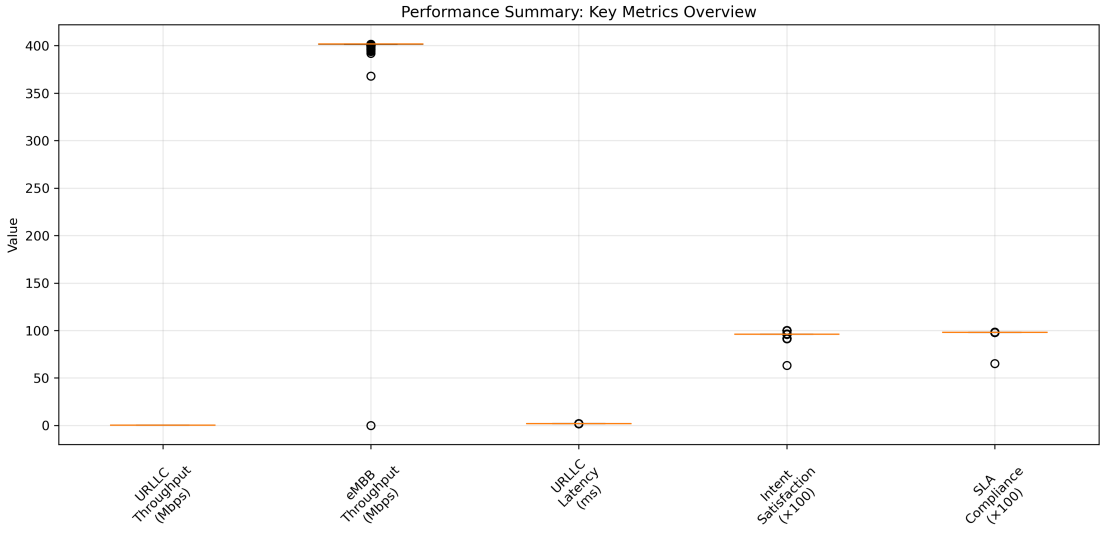


FIGURE 6.3: Boxplot of key KPIs across slices.

Latency Behaviour Latency profiles further reinforce the effectiveness of the agent's decisions. URLLC latency remained below 2 ms for the majority of the simulation, fulfilling stringent low-latency requirements. The eMBB and mMTC slices maintained slightly higher values, around 3–4 ms, still within acceptable ranges for their service categories. These results confirm that the agent respected latency-critical constraints without unnecessarily over-allocating resources.

The per-slice latency over time figure shows URLLC consistently meets its sub-2 ms target, while eMBB/mMTC remain within their tolerance envelopes.

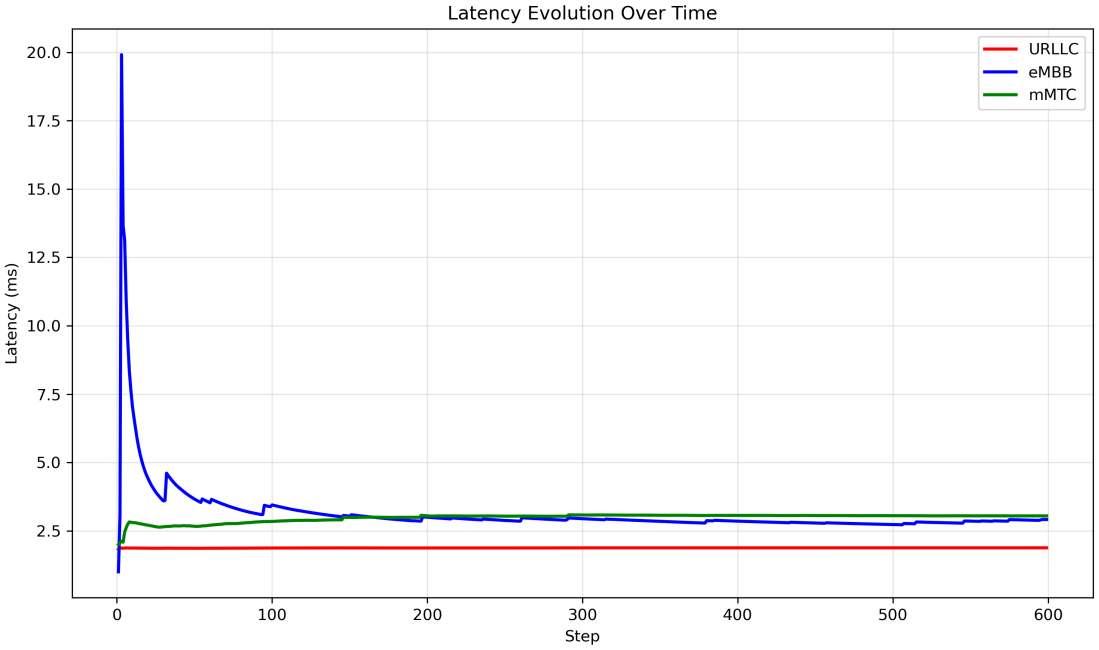


FIGURE 6.4: Per-slice latency over time.

Allocation Patterns The allocation patterns reveal an adaptive policy that dynamically responds to slice demands. While per-step allocations fluctuated, URLLC generally retained a slightly elevated share to preserve latency guarantees, whereas eMBB allocations varied in proportion to its changing traffic demand. The mMTC slice consistently received the smallest share, reflecting its minimal throughput requirements. The allocation histogram shows eMBB clustering around a dominant share, URLLC slightly lower, and mMTC minimal which aligns with the expected traffic priorities.

The following plot presents resource-share trajectories for URLLC, eMBB and mMTC. The policy adapts smoothly while preserving URLLC guarantees and minimising over-provisioning for mMTC.

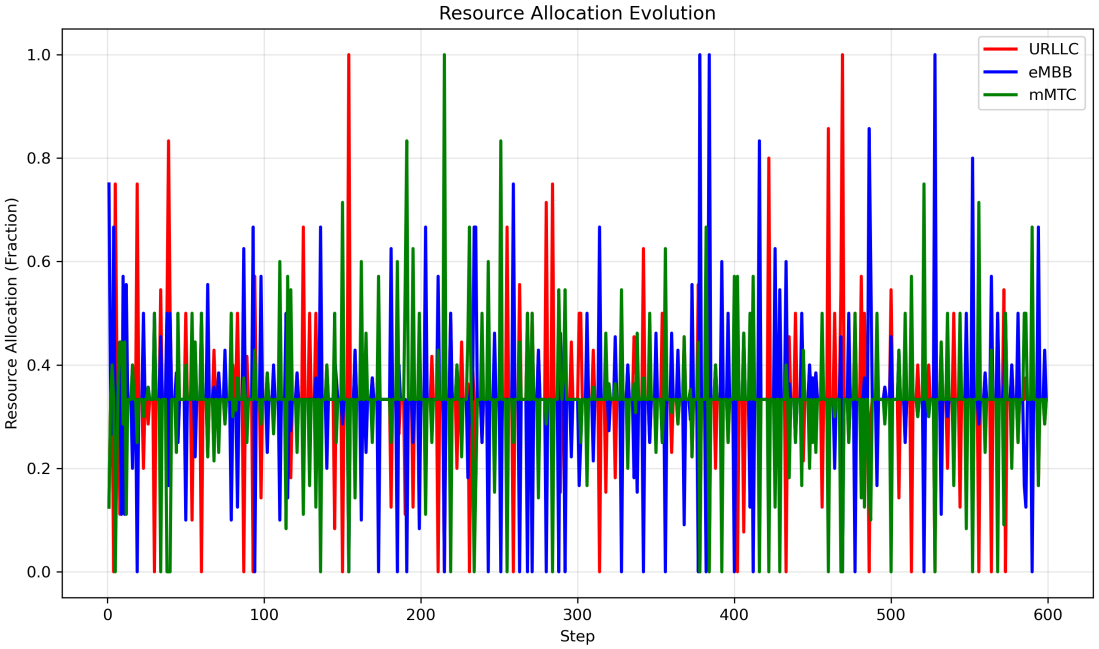


FIGURE 6.5: Resource-allocation decision evolution.

Learning Dynamics and Semantic Performance The learning process exhibited a steep reward increase during the initial episodes, followed by a plateau near the maximum achievable value, indicating that the agent converged to an effective policy early in training. Semantic-layer metrics revealed near-optimal performance, with intent satisfaction consistently high and average QoE exceeding 0.97 for all slices. SLA compliance and network efficiency remained close to saturation, demonstrating the framework's capability to meet contractual guarantees without wasting resources. The per-slice QoE values were nearly identical across service types, confirming that high user-perceived quality was maintained despite differing traffic characteristics.

In this figure the reward rapidly rises and stabilizes, indicating convergence. Moreover, the semantic-aware DQN is shown to outperform baselines on intent, SLA and efficiency.

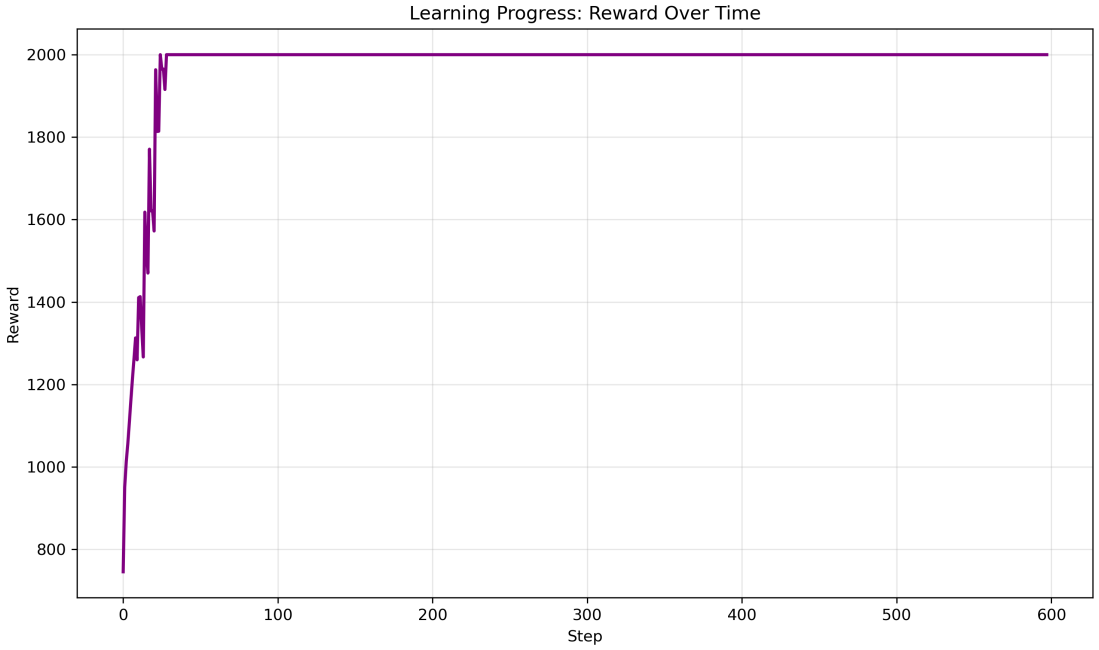


FIGURE 6.6: Reward history

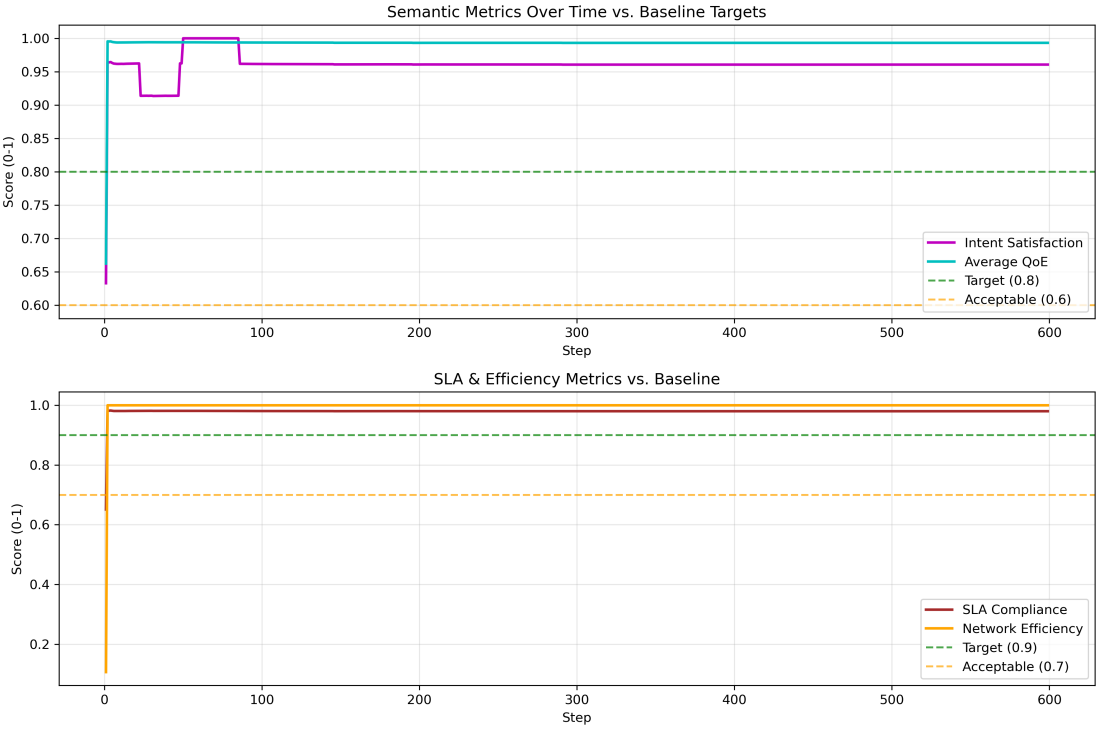


FIGURE 6.7: Semantic vs. baseline

FIGURE 6.8: Learning dynamics and semantic performance.

Correlation of KPIs and Semantic Metrics The correlation between semantic-layer and network performance shows that high throughput for the eMBB slice coincided with equally high semantic scores, demonstrating that maximising low-level KPIs did not come at the

expense of fulfilling service intents. Moreover, training loss remained bounded and stable throughout the experiment, with no signs of divergence. The agent's exploration rate decreased over time through epsilon decay which allowed for effective exploitation of learned strategies while retaining adaptability. The replay memory buffer remained well-utilised resulting to diverse experience sampling.

The next figure represents the correlation between network KPIs and semantic-layer indicators. Strong positive associations confirm that physical-layer optimisation supports semantic objectives.

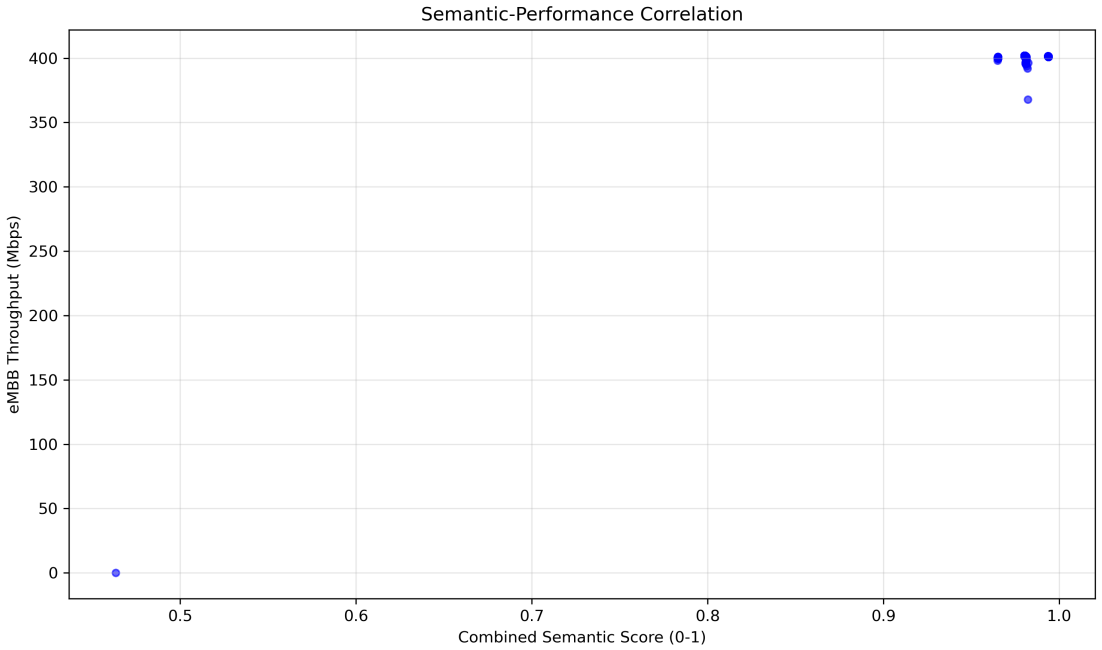


FIGURE 6.9: Correlation matrix between network KPIs and semantic-layer indicators.

Aggregate Evaluation Aggregated KPI visualisations highlight the low variance and consistently high performance of the proposed approach. The boxplot summarising URLLC/eMBB throughput, URLLC latency, intent satisfaction and SLA compliance confirms that all metrics remained in optimal ranges for the majority of the simulation steps. The semantic distribution analysis further shows a dominant concentration above 0.95 for both intent satisfaction and SLA compliance. Overall, the results confirm that the proposed semantic-aware DRL agent effectively balanced resource allocation between slices and delivered high network efficiency with semantic alignment while meeting the strict performance requirements of each service type.

This figure presents the total distributions of chosen semantic KPIs and URLLC latency. The results indicate robust performance due to near optimal values and tight IQRs.

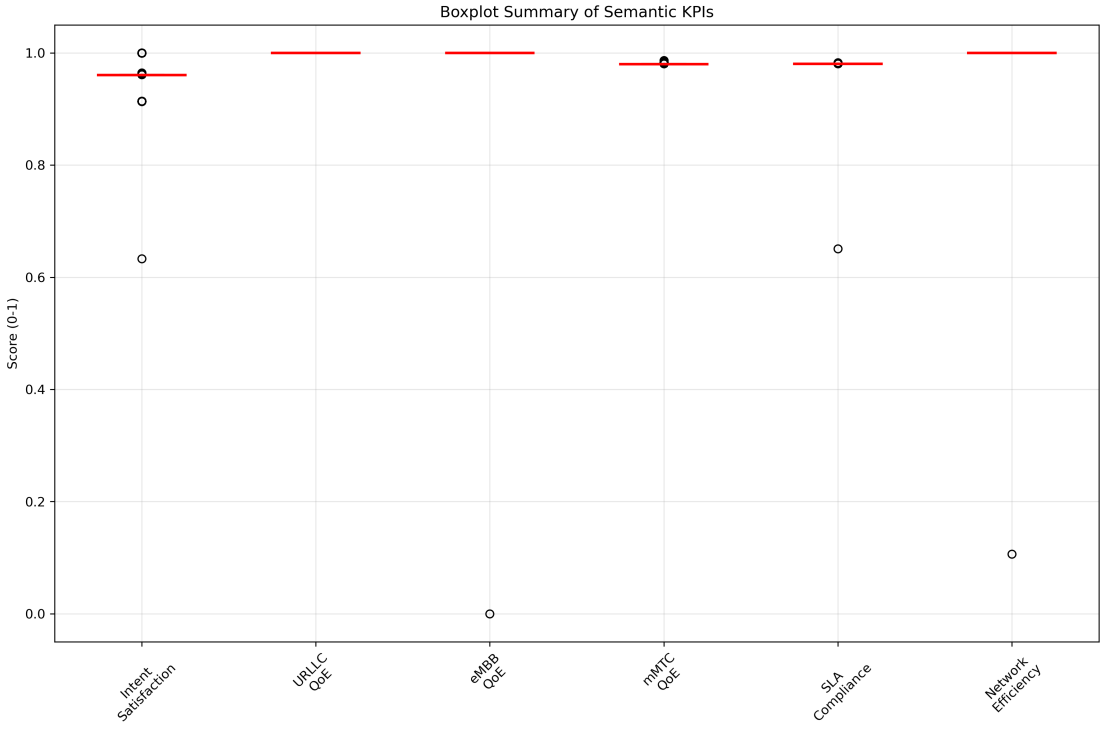


FIGURE 6.10: Aggregate distributions of key semantic KPIs and URLLC latency.

6.4 Discussion of Observed Trends

The results presented in this section's figures highlight several important patterns in the behaviour of the semantic-aware Deep Reinforcement Learning slicing framework.

Firstly, the URLLC slice consistently maintained extremely low latency throughout the simulations (Figure 6.4), demonstrating the agent's capability to prioritise delay-sensitive traffic even under varying demand conditions. This stability highlights the effectiveness of the reward function design, which embedded latency constraints directly into the optimisation objective.

The eMBB slice operated at substantially higher throughput levels but was still able to maintain semantic performance indicators close to their maximum achievable values (Figure 6.9). This outcome suggests that the agent successfully balanced the traditional KPI of throughput with the semantic layer objectives, ensuring that user experience remained high without disproportionately consuming resources.

The mMTC slice maintained minimal resource usage yet achieved high intent satisfaction and SLA compliance. This is also evident in the performance comparison with baseline approaches (Figure 6.7), where the proposed method consistently outperformed static and heuristic allocations across all semantic metrics. This shows that the agent's policy was not

merely allocating resources reactively to observed load but was instead optimising globally across all slices.

Finally, the strong correlation observed between KPI performance and semantic scores suggests that maximising physical-layer performance does not inherently conflict with fulfilling semantic objectives. Instead, the proposed agent design enabled these two goals to be jointly achieved.

6.5 Limitations

While the presented results are promising, certain limitations must be acknowledged. Firstly, the simulation scenario considered a relatively small-scale deployment with only twelve user equipments (UEs). Although this was sufficient to demonstrate the framework's operation, larger and more heterogeneous deployments may introduce additional complexities not captured in this study.

Secondly, the experiments were conducted entirely within a simulated environment, without any hardware-in-the-loop components. As a result, potential real-world factors such as processing delays, hardware-specific performance variations and measurement inaccuracies were not considered.

Lastly, the semantic model integrated into the state representation was tuned specifically for the simulated topology and traffic mix used in this evaluation. Its generalization capability to other network configurations remains to be validated.

6.6 Summary of Experimental Findings

In summary, the proposed semantic-aware DRL slicing framework demonstrated the ability to maintain strict KPI targets while maximizing semantic performance indicators across all network slices. The agent's policy exhibited stability, adaptability to traffic variations and effective resource allocation aligned with service intents. Despite the controlled experimental scope, the results suggest strong potential for applying semantic-aware decision-making to network slicing in future large-scale and heterogeneous deployments.

Chapter 7

Conclusions & Future Work

7.1 Summary of Work

This thesis has presented the conception, development and experimental validation of a semantic-aware Deep Reinforcement Learning (DRL) framework for dynamic slicing in 5G networks. The work was implemented in the `ns-3.44` network simulator, extended with the `NS3-AI` interface to enable real-time interaction between the simulation environment and an external Python-based agent. The official 5G NR module served as the radio access network foundation, providing realistic physical and MAC layer models.

The proposed solution addressed the challenge of aligning resource allocation decisions with both conventional network KPIs and high-level service intents. To this end, semantic-layer indicators such as intent satisfaction, slice-specific QoE, SLA compliance and network efficiency were incorporated into the agent's state representation alongside physical-layer metrics. This integration enabled the learning process to jointly optimise low-level radio performance and user-perceived service quality.

The experimental scenario modeled a deployment with three slice types: URLLC supporting delay-sensitive voice traffic with strict sub-2 ms latency targets, eMBB delivering high-throughput applications such as video and FTP transfers and mMTC handling low-rate IoT telemetry flows. Each slice was configured with distinct traffic models, bandwidth allocations and QoS parameters enforced via NR MAC scheduling. The Deep Q-Network (DQN) agent operated at 100 ms control intervals, dynamically reallocating bandwidth between slices based on the evolving state.

In addition, comparative experiments against static equal-allocation and heuristic demand-proportional allocation baselines demonstrated that the semantic-aware agent maintained KPI stability across all metrics such as throughput, latency and packet loss and

achieved near-optimal semantic performance with intent satisfaction and SLA compliance consistently above 95%, preserved URLLC latency guarantees without over-provisioning low-demand slices and outperformed baselines in balancing resource efficiency with service-specific priorities.

Through this integration of semantic awareness into radio resource management, the framework has shown the feasibility of bridging the gap between Intent-Based Networking principles and physical-layer control. The results confirm that semantic-informed decision-making can enhance both operational efficiency and user-perceived service quality in next-generation networks.

7.2 Key Contributions

The primary contributions of this thesis are centred on the design, implementation and validation of a novel Deep Reinforcement Learning framework with a semantic layer for dynamic slicing in 5G networks. The work advances the state of the art in several important ways.

To begin, a unified architecture was developed that integrates the `ns-3.44` network simulator, the official 5G NR module and the `NS3-AI` interface, allowing for low-latency bidirectional communication between the simulated environment and an external Python-based agent. This architecture allows for the real-time exchange of physical KPIs, semantic indicators and control actions at sub-second intervals.

Moreover, the thesis extends the network slicing decision process beyond conventional physical-layer KPIs by embedding semantic-layer metrics directly into the agent's state representation. metrics enable a richer and more goal-oriented optimization process rather than simply keeping the network to certain hard-coded physical parameters.

Furthermore, the proposed DQN agent was specifically tailored to operate under 5G slicing conditions, with mechanisms for adaptive bandwidth allocation, QoS prioritization and latency-sensitive control. The agent's training incorporated both convergence stability and real-time responsiveness to ensure that learned policies were both effective and deployable in time-sensitive scenarios.

Additionally, a comprehensive experimental framework was established that featured a multi-slice network scenario with realistic URLLC, eMBB and mMTC traffic models. The implementation also included strict QoS enforcement through NR MAC scheduling and baseline comparison against static and heuristic allocation schemes. This setup ensured that performance gains could be objectively attributed to the proposed approach.

Finally, the thesis demonstrates empirically that semantic-aware decision-making can simultaneously achieve high network efficiency and user-perceived quality. Across extensive experiments, the proposed framework maintained URLLC latency targets below 2 ms, sustained eMBB throughput near saturation, preserved mMTC stability and achieved semantic metrics above 95% for the vast majority of simulation steps. These results substantiate the claim that semantic integration into resource allocation can deliver measurable benefits over purely KPI-driven approaches.

7.3 Future Work

While the results presented in this thesis demonstrate the viability and effectiveness of the proposed solution, several avenues remain open for further exploration and refinement.

One promising direction is the extension of the current framework to support larger-scale deployments with increased numbers of gNBs and UEs. Scaling the system in both spatial and traffic dimensions would provide deeper insights into the behavior of semantic-aware resource allocation under higher network load and more diverse mobility patterns. Additionally, integrating heterogeneous radio access technologies, such as millimeter-wave (mmWave) and non-terrestrial networks, could reveal the adaptability of the framework to different propagation and capacity regimes.

Another avenue lies in the exploration of alternative Reinforcement Learning architectures, including policy gradient methods, actor–critic frameworks and multi-agent reinforcement learning. Such approaches could enable more nuanced allocation strategies and improve convergence speed, especially in highly dynamic or partially observable environments.

The semantic-layer design could also be enriched by incorporating additional context-aware indicators. Instances include application-level performance metrics, real-time user mobility predictions and network energy consumption patterns. This would enable the agent to optimize not only for immediate intent satisfaction, but also for long-term operational efficiency and sustainability.

From a system integration perspective, deploying the proposed solution in a hardware-in-the-loop or field-testbed environment would represent an important step toward real-world applicability. This would allow validation under non-ideal conditions, including hardware constraints or imperfect channel state information and control signaling delays. Furthermore, the inclusion of closed-loop interaction with orchestration platforms such as O-RAN or ETSI-compliant network function virtualization (NFV) frameworks, could facilitate seamless integration into modern network management systems.

Finally, the application of the framework to beyond-5G and 6G use cases—such as semantic communications, integrated sensing and communication, or intent-based networking—offers substantial research potential. The principles and mechanisms developed in this thesis are core components of future 6G architecture as described by 3GPP and in the future could be scaled to a 6G semantic-aware solution.

7.4 Final Remarks

This thesis has presented the design, implementation and evaluation of a Deep Reinforcement Learning framework with a semantic layer for dynamic network slicing in 5G environments. By integrating semantic-layer metrics with traditional network KPIs, the proposed approach demonstrated the ability to align resource allocation decisions with both performance objectives and service intents. The experimental results confirmed that the agent maintained high levels of throughput, low latency and strong semantic satisfaction across diverse slice requirements, achieving near-perfect SLA compliance and network efficiency.

The work contributes to the growing body of research on intelligent and intent-driven network management that offers both methodological advancements and practical insights. The combination of semantic-awareness with Reinforcement Learning has shown strong potential for addressing the complexity of next-generation networks, where factors such as heterogeneity, dynamic conditions and diverse service requirements demand adaptable and context-aware control mechanisms.

Looking forward, the techniques developed here can serve as a foundation for future research into scalable and multi-domain resource orchestration. While further work is needed to validate and extend the approach in real-world deployments, the findings of this study reinforce the promise of semantic-aware intelligence as a key enabler for efficient, user-centric and sustainable network operation.

Bibliography

[1] ITU-R, "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," International Telecommunication Union, Tech. Rep. Recommendation ITU-R M.2083-0, Sept. 2015.

[2] 3GPP, "Telecommunication management; Study on management and orchestration of network slicing for next generation network," 3GPP, Tech. Rep. TR 28.801 v15.1.0, Jan. 2018.

[3] S. Wijethilaka and M. Liyanage, "Survey on network slicing for internet of things realization in 5g networks," *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[4] P. Hegde and S. Meena, "A survey on 5g network slicing-epitome and opportunities for a novice," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1–5.

[5] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Network slice lifecycle management for 5g mobile networks: An intent-based networking approach," *IEEE Access*, vol. 9, pp. 80 128–80 146, 2021.

[6] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, pp. 625–655, 2023.

[7] H. O. Otieno, B. Malila, and J. Mwangama, "Deployment and management of intelligent end-to-end network slicing in 5g and beyond 5g networks: A systematic review," *IEEE Access*, vol. 12, pp. 190 411–190 433, 2024.

[8] M. Talal, S. Garfan, R. Qays, D. Pamucar, D. Delen, W. Pedrycz, A. Alamleh, A. Alamoodi, B. Zaidan, and V. Simic, "A comprehensive systematic review on machine learning application in the 5g-ran architecture: Issues, challenges, and future directions," *Journal of Network and Computer Applications*, vol. 233, p. 104041, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804524002182

[9] N. Singh, G. Pranay, and V. Ravi, "Network slicing and traffic classification in 5g networks with explainable machine learning," in *Data Management, Analytics and Innovation*, N. Sharma, A. C. Goje, A. Chakrabarti, and A. M. Bruckstein, Eds. Singapore: Springer Nature Singapore, 2024, pp. 641–658.

[10] A. M. Nagib, H. Abou-Zeid, and H. S. Hassanein, "Transfer learning-based accelerated deep reinforcement learning for 5g ran slicing," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 249–256.

[11] IBM, "Think blog," https://www.ibm.com/think/topics/intent-based-networking, accessed: 2024-08-05.

[12] Ericsson, "Dynamic network slicing," https://www.ericsson.com/en/5g/products/network-slicing, 2023, accessed: 2025-08-05.

[13] H. N. Qureshi, M. Manalastas, S. M. A. Zaidi, A. Imran, and M. O. Al Kalaa, "Service level agreements for 5g and beyond: Overview, challenges and enablers of 5g-healthcare systems," *IEEE Access*, vol. 9, pp. 1044–1061, 2021.

[14] M. M. Rashid, M. Carmen Lucas-Estañ, M. Sepulcre, and J. Gozalvez, "5g ran slicing to support reliability in industrial applications," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4.

[15] J. Rischke, P. Sossalla, S. Itting, F. H. P. Fitzek, and M. Reisslein, "5g campus networks: A first measurement study," *IEEE Access*, vol. 9, pp. 121 786–121 803, 2021.

[16] K. Trichias, A. Kaloxylos, and C. Willcock, "6g global landscape: A comparative analysis of 6g targets and technological trends," in *2024 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2024, pp. 1–6.

[17] V. Gowtham, F. Schreiner, M.-I. Corici, T. Magedanz, I. Zacarias, A. Drummond, and A. Jukan, "Intent-based networking for qos-aware cloud and transport network management based on graph neural networks," in *2023 IEEE Future Networks World Forum (FNWF)*, 2023, pp. 1–7.

[18] J. J. D. Rivera, M. M. S. Sarwar, S. Alam, T. A. Khan, M. Afaq, and W.-C. Song, "An intent-based networking mechanism: A study case for efficient path selection using graph neural networks," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.

[19] S. LaMar, J. J. Gosselin, I. Caceres, S. Kapple, and A. Jayasumana, "Congestion aware intent-based routing using graph neural networks for improved quality of experience in heterogeneous networks," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 477–481.

[20] S. Alam and W.-C. Song, "Intent-based network resource orchestration in space-air-ground integrated networks: A graph neural networks and deep reinforcement learning approach," *IEEE Access*, vol. 12, pp. 185 057–185 077, 2024.

[21] J. Mcnamara, D. Camps-Mur, M. Goodarzi, H. Frank, L. Chinchilla-Romero, F. Cañellas, A. Fernández-Fernández, and S. Yan, "Nlp powered intent based network management for private 5g networks," *IEEE Access*, vol. 11, pp. 36 642–36 657, 2023.

[22] I. Chatzistefanidis, A. Leone, and N. Nikaein, "Maestro: Llm-driven collaborative automation of intent-based 6g networks," *IEEE Networking Letters*, vol. 6, no. 4, pp. 227–231, 2024.

[23] J. Zhang, C. Yang, R. Dong, Y. Wang, A. Anpalagan, Q. Ni, and M. Guizani, "Intent-driven closed-loop control and management framework for 6g open ran," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 6314–6327, 2024.

[24] D. A. Temesgene, E. D. Biyar, A. Silva, A. Likhyani, and A. Zahernszky, "Graph neural network for building prediction agents in intent-based zero-touch networks," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 974–979.

[25] M.-I. Corici, V. Gowtham, T. Magedanz, A. Prakash, and F. Schreiner, "Nemi: A 6g-ready) ai-enabled autonomic network management system for open campus networks," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 674–679.

[26] B. E. Ujcich, A. Bates, and W. H. Sanders, "Provenance for intent-based networking," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 195–199.

[27] X. Luo, H.-H. Chen, and Q. Guo, "Semantic communications: Overview, open issues, and future research directions," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 210–219, 2022.

[28] H. Li, D. Gao, M. Yang, Y. Liang, X. Song, and G. Shi, "A scalable coding method with semantic decomposition for semantic communication," *IEEE Wireless Communications Letters*, vol. 14, no. 7, pp. 2009–2013, 2025.

[29] L. Wang, H. Niu, Z. Lu, and X. Wen, "Demo: A hybrid semantic ran protocol stack design for 6g system and its implementation," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–3.

[30] Y. Tang, N. Zhou, Q. Yu, D. Wu, C. Hou, G. Tao, and M. Chen, "Intelligent fabric enabled 6g semantic communication system for in-cabin scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1153–1162, 2023.

[31] N. H. Mahmood, H. Alves, O. A. López, M. Shehab, D. P. M. Osorio, and M. Latva-Aho, "Six key features of machine type communication in 6g," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, 2020, pp. 1–5.

[32] A. Bouroudi, A. Outtagarts, and Y. Hadjadj-Aoul, "Dynamic machine learning algorithm selection for network slicing in beyond 5g networks," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, 2023, pp. 314–316.

[33] M. A. Islam Arif, S. Kabir, M. F. Hussain Khan, S. Kumar Dey, and M. M. Rahman, "Machine learning and deep learning based network slicing models for 5g network," in *2022 25th International Conference on Computer and Information Technology (ICCIT)*, 2022, pp. 96–101.

[34] S. Kukliński, L. Tomaszewski, R. Kołakowski, and P. Chemouil, "6g-lego: A framework for 6g network slices," *Journal of Communications and Networks*, vol. 23, no. 6, pp. 442–453, 2021.

[35] N. Zohar, "Beyond 5g: Reducing the handover rate for high mobility communications," *Journal of Communications and Networks*, vol. 24, no. 2, pp. 154–165, 2022.

[36] N. Nonaka, K. Muraoka, T. Okuyama, S. Suyama, Y. Okumura, T. Asai, and Y. Matsumura, "28 ghz-band experimental trial at 283 km/h using the shinkansen for 5g evolution," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[37] C. V. Nahum, L. De Nóvoa Martins Pinto, V. B. Tavares, P. Batista, S. Lins, N. Linder, and A. Klautau, "Testbed for 5g connected artificial intelligence on virtualized networks," *IEEE Access*, vol. 8, pp. 223 202–223 213, 2020.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[39] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5g radio access network slicing," 2020. [Online]. Available: https://arxiv.org/abs/2009.06579