



**Frankfurt University of
Applied Sciences**

– Faculty 2: Computer Science and Engineering –

**Artificial Intelligence Techniques for Creating,
Administering and Securing Computer Networks**

Thesis submitted in order to obtain the academic degree

Master of Science (M.Sc.)

submitted on 04. June 2025

Phuoc Huy Lam

Matriculation Number: 1104785

First Supervisor : Prof. Dr. Christian Baun
Second Supervisor : Prof. Dr. Thomas Gabel

Declaration

Hereby I assure that I have written the presented thesis independently and without any third party help and that I have not used any other tools or resources than the ones mentioned/referred to in the thesis.

Any parts of the thesis that have been taken from other published or yet unpublished works in terms of their wording or meaning are thoroughly marked with an indication of the source.

All figures in this thesis have been drawn by myself or are clearly attributed with a reference.

This work has not been published or presented to any other examination authority before. I am aware of the importance of the affidavit and the consequences under examination law as well as the criminal consequences of an incorrect or incomplete affidavit.

Frankfurt, 04. June 2025

PHUOC HUY LAM

Phuoc Huy Lam

Contents

1	Introduction	2
1.1	Background and context	2
1.2	Motivation for research	2
1.3	Research Objectives	3
1.4	Scope and Limitations	3
1.4.1	Scope	3
1.4.2	Limitations	3
1.5	Structure of the Thesis	4
2	Fundamentals of AI in Computer Network	5
2.1	Basics of Artificial Intelligence	5
2.2	Overview of AI in Computer Networking	8
2.3	State of the art AI Techniques used in Computer Networking	10
2.4	Notable Tools and Frameworks	11
2.4.1	Cisco DNA Center	11
2.4.2	TensorFlow-Agents in Custom SDN Controllers	12
2.4.3	Red Hat Ansible Lightspeed	12
3	AI Techniques for Network Creation	14
3.1	Network Topology Optimization	14
3.1.1	Reinforcement Learning (RL):	14
3.1.2	Genetic Algorithms (GAs)	16
3.2	Network Planing	17
4	AI Techniques for Network Administration	19
4.1	Network Traffic Prediction	19
4.1.1	Machine Learning Approaches	19
4.1.2	Deep Learning Approaches	20
4.2	Predictive Maintenance	22
5	AI Techniques for Network Security	24
5.1	Threat Detection and Anomaly Detection with AI	24

5.2 Ethical Implications and Challenges in AI-Driven Security	26
6 AI techniques Implementation	27
6.1 Methodology and Development Environment Setup	27
6.2 SDN Topology Optimization Using Reinforcement Learning	28
6.2.1 Project Workflow	28
6.2.2 Deployment and Execution Instructions	28
6.3 Smart Network Traffic Classification Using Machine Learning	32
6.3.1 Project Workflow	32
6.3.2 Deployment and Execution Instructions	33
7 Conclusion and Future Work	38
A Programmcode	40
Bibliography	47

Abstract

As computer networks continue to grow in size and complexity, driven by technologies such as Cloud Computing, Internet of Things (IOT), 5G, and Edge Computing, managing them effectively has become increasingly challenging. Traditional rule-based and manual approaches are no longer sufficient to ensure the efficiency, scalability, and resilience required by today's dynamic and high-demand environments. In this context, Artificial Intelligence (AI) offers new opportunities to build smarter, more adaptive, and more secure network systems.

This thesis explores how AI techniques can be applied to three key areas of computer networking: creating network architectures, managing ongoing operations, and improving network security. The research combines a review of existing literature with practical experiments to understand how methods like Machine Learning (ML), Deep Learning (DL), and Reinforcement Learning (RL) can help solve real-world networking problems. These include optimizing network topologies, predicting traffic patterns, detecting anomalies, and automating security responses.

The results show that AI has the potential to significantly improve how networks are designed, operated, and protected. While there are still challenges, such as limited datasets and computing resources, the study demonstrates that AI-driven solutions can make networks more efficient, reliable, and responsive.

Keywords: Artificial Intelligence, AI, AI Techniques, Network, Computer Networking, Network Creation, Network Administration, Network Security.

Chapter 1

Introduction

1.1 Background and context

Computer networks are the fundamental infrastructure for communication, data sharing, and service delivery across industries in today's society. As these systems become more complex each year, ensuring their efficiency, security, and reliability becomes increasingly difficult. Traditional approaches to network administration and security often struggle to keep pace with emerging threats and the expanding number of connected devices in dynamic environments.

Advancements in hardware and software have enabled the application of intelligent computational methods to networking. Techniques such as ML, Neural Network (NN), DL, and Natural Language Processing (NLP) are being explored to address key issues, including network topology optimization, network traffic prediction, anomaly detection, and threat mitigation.

In this thesis, we explore the application of AI techniques in the design, management, and security of computer networks through both academic review and practical implementation. We investigate how these methods can be integrated into modern network infrastructures to improve operational efficiency and strengthen security.

1.2 Motivation for research

The rapid expansion of cloud computing and the ongoing global shift toward digital transformation have introduced significant complexity into modern computer networks. As a result, traditional network management methods that rely on manual configurations and reactive security measures are increasingly inadequate. Key limitations of these conventional approaches include:

- **Scalability Problems:** Manual procedures by engineers often can't keep up with the needs of modern network infrastructures, leading to deployment delays and configuration mistakes.
- **Incomplete Visibility:** Legacy tools frequently lack comprehensive visibility across hybrid and multi-cloud infrastructures, making it harder to monitor traffic and identify issues.
- **Reactive Security Model:** Traditional security models often respond only after threats are identified, rather than being designed to stop them in advance. This approach leads to damage control rather than prevention.
- **High Costs:** Maintaining and troubleshooting legacy systems is resource-intensive, prone to human error, and associated with high operational costs.

To overcome these challenges, today's networks require solutions that can scale easily, adapt to changing conditions, and respond to problems before they escalate. AI technologies powered by data and automation, particularly those that utilize intelligent decision-making, offer a practical approach. They can help make network operations more efficient, accurate, and responsive in real time.

1.3 Research Objectives

The primary objectives of this research are:

- Explore state-of-the-art AI techniques and their current applications in computer networking through a comprehensive review of academic literature.
- Identify potential use cases for AI in various aspects of computer network operations.
- Analyze the effectiveness of AI-based approaches for the design, management, and security of computer networks.
- Implement and evaluate selected AI-driven solutions in practical networking scenarios to assess their performance and impact.

1.4 Scope and Limitations

1.4.1 Scope

This thesis investigates the application of AI techniques in three core areas of computer networking:

- **Network Creation:** Exploring AI-driven approaches to network planning and topology optimization.
- **Network Administration:** Applying AI algorithms for network traffic prediction and predictive maintenance.
- **Network Security:** Using AI methods for threat detection, anomaly identification, and automated response strategies.

The research includes a review of existing AI-based networking solutions, followed by practical implementation and evaluation of selected AI techniques in simulated environments.

1.4.2 Limitations

Given the scope and time frame of this thesis project, several limitations must be acknowledged:

- **Time Constraints:** The research is conducted within a limited period, which restricts the extent of experimentation and the depth of analysis.
- **Data Availability:** The study primarily uses publicly available data sets and small-scale simulations, rather than extensive real-world deployments.
- **Implementation Scope:** The AI models developed in this research are intended as proof-of-concept implementations, not as production-ready solutions.
- **Computational Resources:** The development and testing of AI techniques are subject to limitations in available hardware and computing infrastructure.

Despite these constraints, the study aims to offer meaningful insights into the applicability and potential benefits of AI in modern network management and security.

1.5 Structure of the Thesis

This section provides an overview of the thesis structure, outlining the content and objectives of each chapter to guide the reader through the research.

- **Chapter 1: Introduction.** Introduces the research topic, its motivation, and the main objectives. It discusses the challenges faced in modern computer networks and outlines the potential of AI in addressing these issues. The chapter also defines the scope, methodology, and limitations of the study.
- **Chapter 2: Fundamentals of AI in Computer Networking.** Presents foundational concepts in AI relevant to networking, including ML, DL, NN and RL. It also explains how these techniques align with network functions such as topology optimization, fault detection, and threat mitigation, and identifies key integration points for AI in network systems.
- **Chapter 3: AI Techniques for Network Creation.** Examines the use of AI in the design and deployment of computer networks. Topics include network topology optimization and network planning.
- **Chapter 4: AI Techniques for Network Administration.** Explores how AI can improve network management through automation and intelligent decision-making. It focuses on techniques such as traffic prediction and predictive maintenance.
- **Chapter 5: AI Techniques for Network Security.** Investigate the role of AI in enhancing network security. This includes threat detection, anomaly identification, and intrusion prevention. The chapter also considers the ethical implications and limitations of using AI for security purposes.
- **Chapter 6: Implementation and Discussion.** This chapter explores the practical implementation of selected AI techniques across three key focus areas. Each solution is designed to run locally within a simulated environment, demonstrating the feasibility and effectiveness of AI in improving network performance, rather than producing production-level code. The chapter further analyzes the outcomes of these implementations, assessing their significance and the broader impact of AI on computer networking.
- **Chapter 7: Conclusion.** Provides a summary of the thesis's key findings and contributions in AI-driven network design, management, and security. It discusses the research limitations, highlights practical and theoretical implications, and proposes avenues for future investigation in the field.

Chapter 2

Fundamentals of AI in Computer Network

2.1 Basics of Artificial Intelligence

According to the Oxford English Dictionary, AI is defined as: “The capacity of computers or other machines to exhibit or simulate intelligent behaviour; the field of study concerned with this. In later use also: software used to perform tasks or produce output previously thought to require human intelligence, especially by using machine learning to extrapolate from large collections of data” [7].

This chapter introduces the key ideas and technologies that make up the modern AI landscape. Since this thesis focuses on how AI can be applied in the real world, specifically in computer networks, we will not go deep into the theory or complex math behind these systems. Instead, the goal is to provide a straightforward overview to help the reader understand how AI works at a basic level and how it can be used effectively in networking environments.

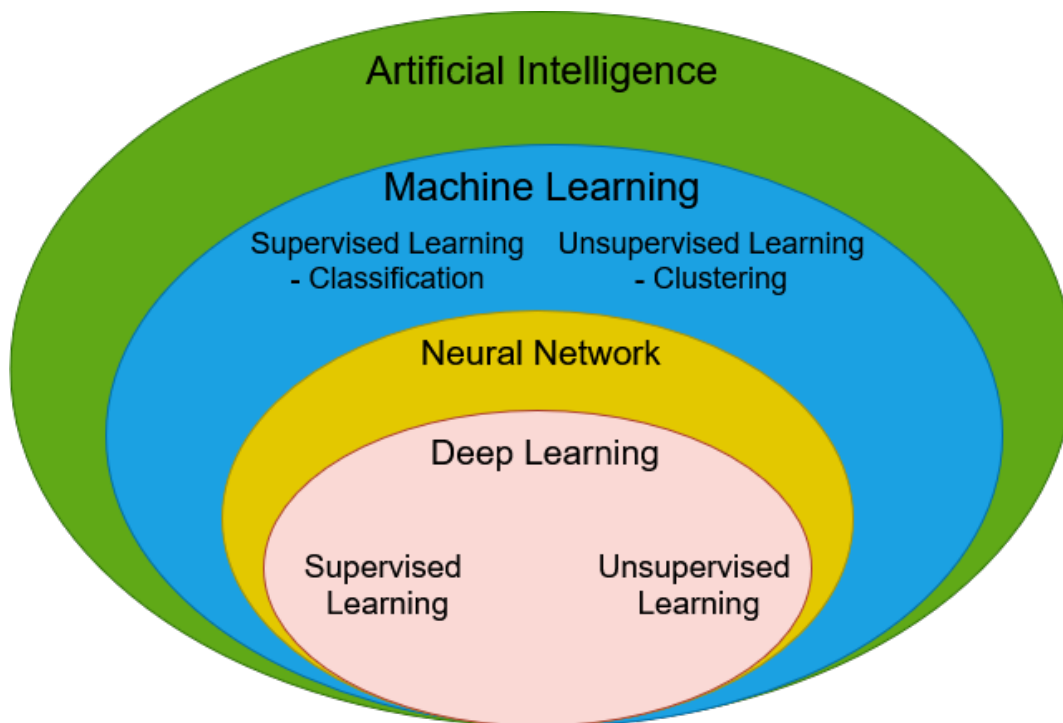


Figure 2.1: Core AI concepts and their relationship, as mentioned in [17]

In the paper “A Review of Machine Learning and Deep Learning Applications” [17], the authors explore the foundational concepts and practical applications of AI, with a particular focus on ML and DL. Figure 2.1 presents a visual representation of the core techniques within AI and illustrates the hierarchical relationship among them.

In the outer layer is **Artificial Intelligence (AI)**, which broadly refers to the simulation of human intelligence by computer systems. AI involves various cognitive functions, including learning (acquiring knowledge and understanding how to apply it), reasoning (inferring from data or predefined rules), and self-correction (improving performance over time based on feedback).

The next layer is **Machine Learning (ML)**, a subset of AI that enables computer systems to learn from data and improve their performance over time without human intervention. Rather than relying on fixed instructions, ML algorithms identify patterns within data, allowing systems to make informed decisions or predictions. This adaptability makes ML particularly useful for solving complex problems such as image recognition, natural language processing, and predictive analytics.

As ML systems process more data, they refine their internal models through feedback, gradually increasing their accuracy and efficiency. ML algorithms generally fall into two main categories:

- **Supervised Learning:** In this approach, the model is trained on a labeled dataset, where each input is associated with a known output. The goal is to learn a mapping from inputs to outputs in order to make accurate predictions on new, unseen data. Supervised learning is commonly used for tasks such as classification and regression.
- **Unsupervised Learning:** This method involves working with data that has no predefined labels. The goal is to uncover hidden patterns, groupings, or structures within the dataset. Typical applications include clustering, anomaly detection, and dimensionality reduction.

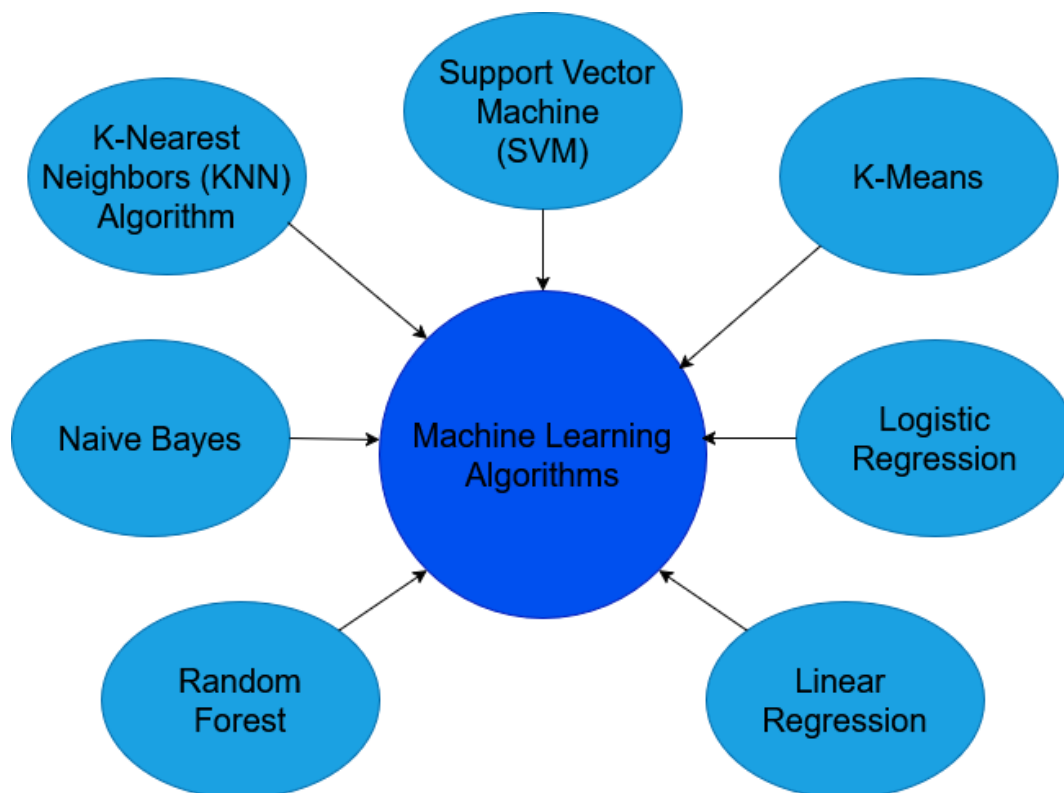


Figure 2.2: Some of the most commonly used machine learning algorithms. Source: [4]

The third layer is **Neural Networks (NN)**. A neural network is a type of ML model that mimics the way the human brain processes information to make decisions. They are composed of layers of interconnected nodes, or “neurons,” which transmit data through the network in stages. During training, the connections between these neurons are assigned weights,

which are adjusted to reduce errors and improve the model's performance. Their ability to learn from experience and generalize to new data has established neural networks as a key component of modern AI.

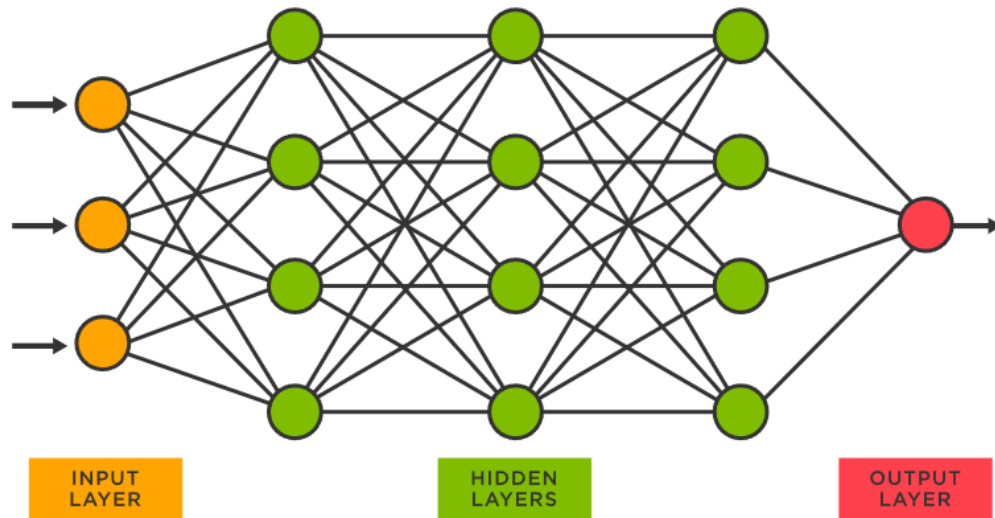


Figure 2.3: Example of a Neural Network. Source: [18]

Figure 2.3 illustrates an example of a neural network structure. It is normally divided into 3 layers:

- **Input Layer:** The starting point of the neural network where data is introduced into the model. Each neuron in this layer corresponds to one feature of the input data.
- **Hidden Layers:** These are the intermediate layers between the input and output layers. They carry out the majority of the computations, with each neuron processing inputs and passing the results to the next layer. Neural networks can have one or many hidden layers, depending on the complexity of the problem.
- **Output Layer:** The final layer of the network that produces the result or prediction. The number and type of neurons in this layer vary based on the task, such as using a single neuron for binary classification or multiple neurons for multi-class classification or regression.

Deep Learning (DL) is a specialized subfield of ML that utilizes neural networks with multiple layers, often referred to as deep neural networks, to learn from large and complex datasets. These models are particularly effective at capturing intricate patterns in data, making them well-suited for tasks such as image recognition, speech processing, and natural language understanding. One of the key advantages of deep learning is its ability to automatically extract relevant features from raw data, reducing the need for manual feature engineering. This capability has significantly contributed to the development of highly accurate and efficient AI systems. Similar to ML, DL approaches can be broadly classified into:

- **Supervised Learning:** In this approach, models are trained on labeled datasets, learning to map inputs to their corresponding outputs. It is commonly used for classification and regression tasks.
- **Unsupervised Learning:** A deep learning approach where models learn patterns and structures from unlabeled data without predefined outputs, often used for tasks like clustering and feature extraction.

In addition to the aforementioned techniques, **Reinforcement Learning (RL)** is another widely adopted approach in the field of AI. RL involves an agent that learns to make optimal decisions through interaction with its environment. Through a process of trial and error, the agent takes actions and receives feedback in the form of rewards or penalties. Over time, it learns to optimize its behavior to maximize cumulative rewards. Based on behavioral psychology, RL is particularly effective in domains such as robotics, game playing, and autonomous systems. Unlike supervised learning, RL does not require labeled datasets, as learning is driven by trial-and-error experiences.

2.2 Overview of AI in Computer Networking

In recent years, the field of AI has experienced a period of rapid growth and innovation, often referred to as an ongoing "AI Spring" [8]. This surge has been driven by the emergence of high-profile models such as ChatGPT and Claude, which fall under the category of Generative Artificial Intelligence (GENAI). Generative AI refers to a class of technologies capable of producing original content, such as text, images, audio, and video, by learning patterns from large-scale datasets. Unlike traditional systems that rely on predefined rules, generative models employ DL techniques to generate outputs that closely resemble those created by humans.

One of the key drivers of generative AI's accelerated progress and adoption is the decreasing cost of training and deploying advanced models (as illustrated in Figure 2.4). This trend has been enabled by advancements in both hardware and software. On the hardware side, more powerful Graphics Processing Units (GPUs) and specialized AI accelerators, such as those developed by NVIDIA, have greatly improved computational efficiency. On the software side, the development of more efficient algorithms and optimization of deep learning frameworks has significantly reduced training time and resource requirements.

As a result, modern generative models are becoming more capable of addressing complex challenges while remaining cost-effective and scalable. These tools are now widely applied across industries for tasks such as content generation, design support, data analysis, and decision-making. Major technology companies are heavily investing in this area, for instance, Microsoft's integration of Copilot into its software suite and Google's continued development of its Gemini platform, further reinforcing the mainstream adoption of generative AI technologies.

Inference price across select benchmarks, 2022–24

Source: Epoch AI, 2025; Artificial Analysis, 2025 | Chart: 2025 AI Index report

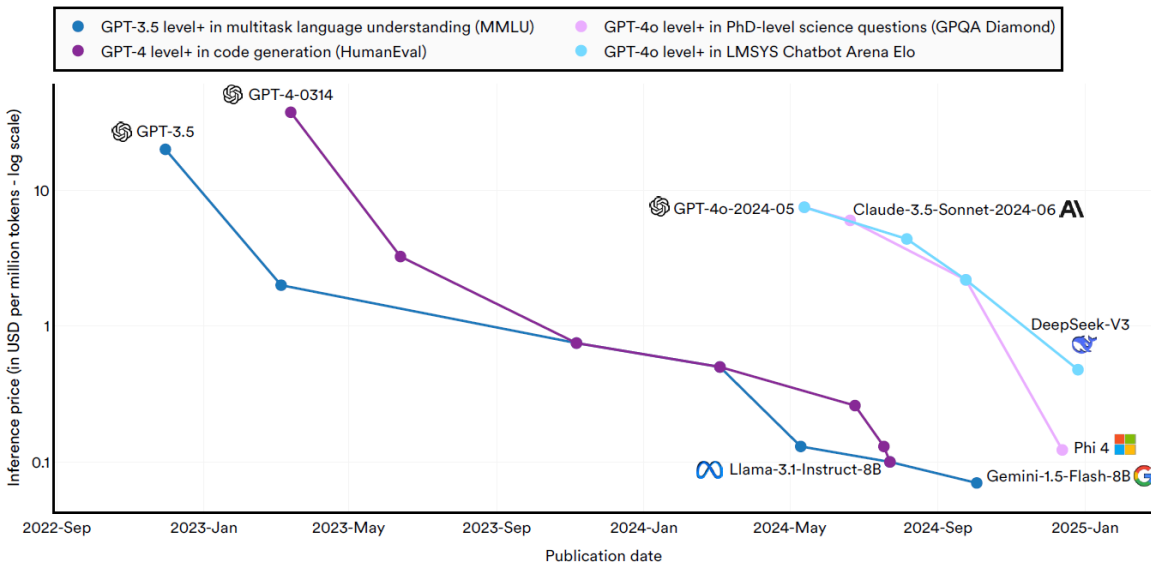


Figure 1.3.22

Figure 2.4: The inference cost of AI is going down. Source: [9]

Alongside its broader impact on the technology landscape, AI has also made significant contributions to the field of computer networking. AI-driven techniques are transforming core aspects of network design and operation, enabling advanced capabilities in automation, resource optimization, and cybersecurity. By reducing reliance on manual intervention, these technologies improve the efficiency, adaptability, and resilience of network systems. As a result, AI is fundamentally reshaping how modern networks are planned, managed, and secured.

The integration of AI into computer networks has become a major focus in both academic research and industry practice. With the rapid advancement of technologies such as 5G, Edge Computing, and the IOT, today's network infrastructures are increasingly dynamic, complex, and heterogeneous. These evolving conditions demand more intelligent, data-driven

approaches to network management. Figure 2.5 displays the key domains where AI can be effectively applied to enhance the performance, reliability, and security of contemporary computer networks.

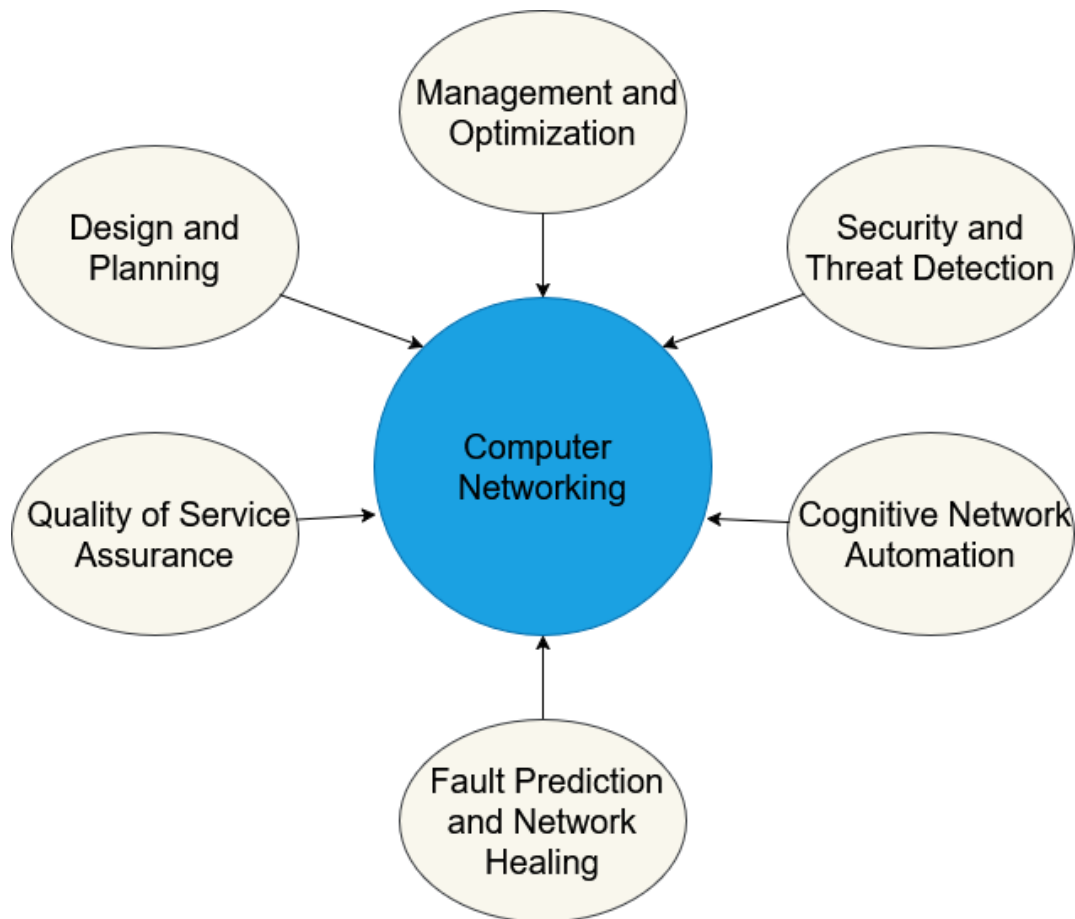


Figure 2.5: Integration points for AI in Computer Networking

- **Network Management and Optimization:** AI has shown significant potential in improving network management, an area where traditional rule-based systems often fall short due to the scale and speed of modern network traffic. Machine learning algorithms, particularly those involving supervised, RL, and DL, enable real-time network monitoring, traffic prediction, and dynamic bandwidth allocation. These techniques allow for the detection of traffic patterns, congestion forecasting, and adaptive routing. Such capabilities are especially effective in Software-Defined Networking (SDN) environments, where AI can assist controllers in making intelligent, data-driven decisions to optimize network resource utilization.
- **Security and Threat Detection:** One of the most prominent uses of AI in networking is in cybersecurity. AI models can be trained to detect anomalies in network behavior, enabling the identification of emerging threats such as zero-day attacks and Advanced Persistent Threats (APTs). Unsupervised learning methods, including clustering and autoencoders, are commonly used in Intrusion Detection and Prevention Systems to identify suspicious activities without prior knowledge of threat signatures. Moreover, AI can automate incident response processes, minimizing reaction times and enhancing overall system resilience.
- **Quality of Service Assurance:** Ensuring consistent Quality of Service (QoS) is critical for latency-sensitive applications such as video streaming, online gaming, and Voice over Internet Protocol (VoIP). AI enhances QoS by dynamically managing parameters like latency, jitter, and packet loss through predictive analytics and intelligent

feedback loops. Reinforcement learning, in particular, can be employed to continuously optimize QoS strategies in real time, helping to meet Service-Level Agreements (SLAs) with minimal manual intervention.

- **Fault Prediction and Network Healing:** AI enables predictive maintenance by analyzing historical network performance data to anticipate failures before they occur. This enables proactive measures, such as rerouting traffic, isolating malfunctioning segments, or initiating automated recovery protocols. These predictive and self-healing capabilities contribute to increased network uptime, improved reliability, and reduced operational costs.
- **Network Design and Planning:** AI also plays a significant role in the early stages of network development. Using optimization techniques and evolutionary algorithms, AI can simulate and evaluate various network topologies, assist with capacity planning, and streamline configuration management. Additionally, AI models can factor in external variables, such as power consumption, physical constraints, and future traffic demands, to create more adaptive, scalable, and cost-effective network architectures.
- **Cognitive Network Automation:** The long-term goal of AI integration in networking is the development of cognitive networks, intelligent systems that are capable of self-management, self-optimization, and self-protection. These networks use AI to continuously monitor their environment, learn from both historical and real-time data, and autonomously adjust their behavior to maintain optimal performance and security.

As these technologies continue to mature, the integration of AI into computer networking is expected to progress toward fully autonomous, self-sustaining systems. This evolution marks a significant advancement in communication infrastructure, offering the potential for networks that are not only more efficient but also more resilient and responsive to dynamic conditions.

2.3 State of the art AI Techniques used in Computer Networking

State-of-the-art AI techniques are being increasingly adopted across various layers of modern network architectures, including traffic management, resource allocation, intrusion detection, and network optimization. This section highlights key examples of how these advanced methods are applied in current networking practices. Each of the three core categories (network creation, network administration, and network security) will be explored in more depth in the following chapters.

- **Machine Learning for Network Management:** Both Supervised and Unsupervised ML techniques are widely used to automate networking tasks such as traffic classification, anomaly detection, and performance forecasting. Supervised learning algorithms, such as Decision Trees, Support Vector Machines (SVM), and Random Forests, are commonly used to classify network flows and detect intrusions using labeled datasets. In contrast, unsupervised methods like K-means Clustering and Principal Component Analysis (PCA) are applied to uncover abnormal traffic patterns or emerging security threats without the need for labeled data. A detailed survey by Boutaba et al. [6] provides a comprehensive review of machine learning applications in network management, categorizing techniques across various networking domains.
- **Deep Learning for Traffic Analysis and Intrusion Detection:** DL has proven especially effective in network traffic analysis and intrusion detection. Convolutional Neural Networks (CNNs) are well-suited for extracting spatial features from network traffic data, while Recurrent Neural Networks (RNNs) are adept at capturing temporal dependencies, making them ideal for time-series tasks like traffic prediction and anomaly detection. These models often outperform traditional methods, particularly in high-volume, dynamic network environments. Abbasi et al. [1] present a comprehensive survey of deep learning techniques for Network Traffic Monitoring and Analysis (NTMA), covering key methodologies, practical applications, and ongoing research challenges.
- **Reinforcement Learning for Dynamic Resource Allocation:** RL enables agents to learn adaptive strategies for tasks such as routing, bandwidth management, and congestion control by interacting with their environment and receiving feedback in the form of rewards. Deep Reinforcement Learning (DRL), which integrates RL with Deep Neural Network (DNN), extends these capabilities to complex, high-dimensional scenarios. For example, Koo et al. [12] propose a DRL-based framework for dynamic resource allocation in network slicing, addressing heterogeneous

service requirements and time-varying traffic conditions. Their study demonstrates that the DRL approach achieves superior adaptability, efficiency, and scalability compared to traditional methods, making it a promising solution for modern network management challenges.

- **AI in Software-Defined Networking and Network Function Virtualization:** AI is also being integrated into SDN and Network Function Virtualization (NFV) to enhance programmability, flexibility, and automation. In SDN, AI algorithms analyze real-time network conditions to dynamically adjust routing strategies and optimize traffic distribution. In the context of NFV, AI supports the efficient deployment, scaling, and orchestration of Virtualized Network Functions (VNF), leading to improved resource utilization and reduced operational complexity. These developments are key enablers of self-managing, intent-driven networks, which represent a step toward fully autonomous infrastructures. A detailed survey on ML applications in SDN is provided by Xie et al. [22].

2.4 Notable Tools and Frameworks

The integration of AI into computer networking has led to the development of various tools and frameworks that enable intelligent automation, performance optimization, and security enhancement. These platforms support advanced capabilities such as traffic management, anomaly detection, predictive maintenance, and automated policy enforcement. This section provides an overview of some of the most prominent tools and frameworks that apply AI techniques to networking, highlighting their main features, functionalities, and common use cases.

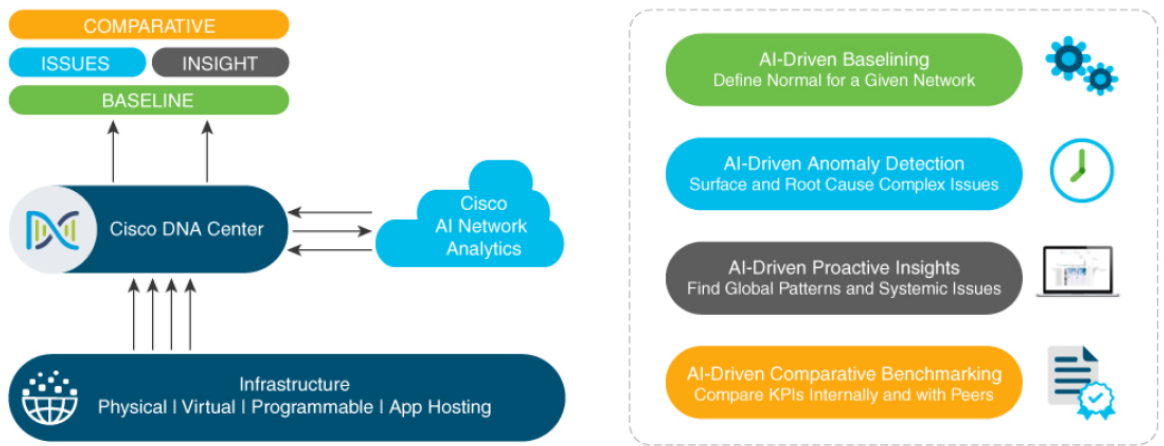


Figure 2.6: Overview of Cisco DNA Center's AI-driven capabilities. Source: [19]

2.4.1 Cisco DNA Center

Cisco's Digital Network Architecture (DNA) Center is a centralized network management platform designed to simplify enterprise network operations through automation and AI-driven insights. It integrates ML and data analytics to support intelligent decision-making and reduce the operational complexity of managing large-scale networks.

One of the core features of Cisco DNA Center is **Network Assurance**, which uses AI to continuously monitor network health and performance. By collecting and analyzing real-time telemetry data from network devices, it can identify anomalies, detect root causes of issues, and suggest actionable recommendations. The platform also includes **AI-powered predictive analytics**, which helps forecast potential failures or congestion before they occur, allowing administrators to take proactive measures.

Another key capability is **Intent-based Networking**, where network configurations are aligned with high-level business goals. Administrators can define intent (such as access policies or application priorities), and the system translates this

into automated configurations across the network infrastructure. Overall, Cisco DNA Center exemplifies how the industry has leveraged AI and automation to create smarter, more responsive, and more secure network environments.

2.4.2 TensorFlow-Agents in Custom SDN Controllers

In academic and research environments, there's growing interest in combining RL with SDN to make networks more adaptive and intelligent. One of the popular tools used for this purpose is **TensorFlow-Agents**, a flexible library built on top of TensorFlow that simplifies the development of RL models.

TensorFlow-Agents is often used to build RL agents that interact with SDN controllers, allowing them to learn how to manage and optimize network operations automatically. These agents don't rely on fixed rules. Instead, they learn from experience by receiving feedback from the network environment, adjusting their strategies over time. Common use cases include Dynamic Routing, Congestion Control, and QOS enforcement.

As demonstrated by Xu et al. [23], this approach is especially popular in experimental setups like simulated network testbeds. These controlled environments give researchers the freedom to test new ideas and fine-tune their models before applying them in real-world scenarios. For instance, a RL agent can be trained to reroute traffic based on real-time congestion, ultimately making the network more efficient and responsive.

One of the strengths of TensorFlow-Agents is its flexibility. It supports different types of learning algorithms and can be easily integrated with SDN platforms like OpenFlow. Researchers can build custom policies, simulate various network conditions, and experiment with how different learning strategies perform under stress.

Figure 2.7 gives a visual example of how reinforcement learning works using TensorFlow-Agents. The agent observes the state of the network, takes actions (such as changing routes), and receives feedback in the form of rewards, which help it learn better decisions over time. TensorFlow-Agents offers a powerful and accessible way to explore how AI can make networks smarter. It helps bridge the gap between theoretical research and practical applications in intelligent network control.

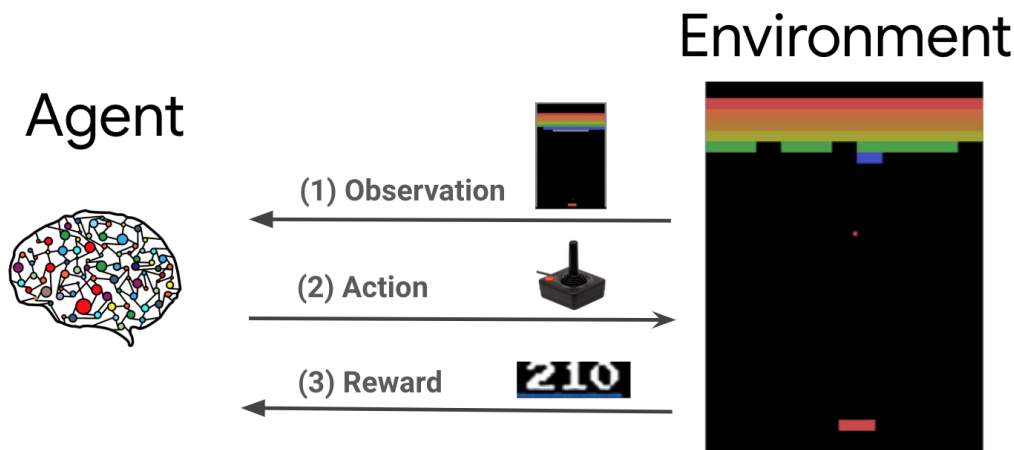


Figure 2.7: An example of Reinforcement Learning with TensorFlow Agents. Source: [3]

2.4.3 Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed is an advanced extension of the Ansible automation platform, developed in collaboration with IBM's Watson Code Assistant. Its purpose is to bring the power of generative AI into the world of IT and network automation, helping users create, maintain, and scale automation workflows more efficiently.

Ansible Lightspeed's core function is translating natural language input into structured Ansible YAML code. This means that users can describe what they want to achieve in plain language, and the platform will generate the corresponding

automation script. For example, instead of writing an entire playbook manually, a user might simply type “Configure VLAN on Cisco switches and apply ACLs” and Lightspeed will produce the YAML code needed to execute that action.

This approach dramatically reduces the time and expertise needed to build automation routines, especially for engineers who may not be deeply familiar with Ansible’s syntax or structure. It also helps avoid human errors often introduced in manual scripting. As users interact with the platform, the underlying AI model improves over time, learning from community best practices and user feedback to provide increasingly accurate and context-aware code suggestions.

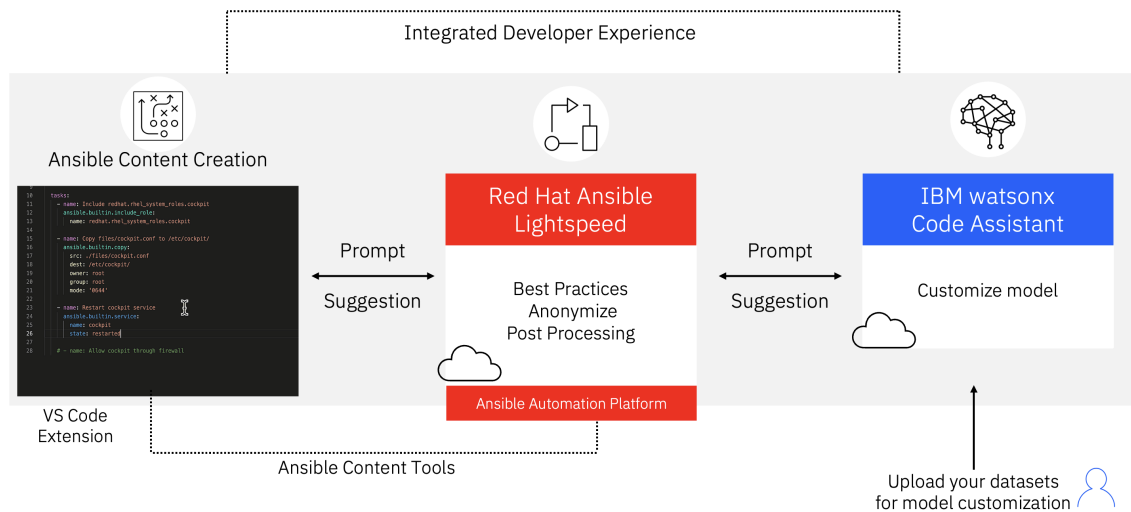


Figure 2.8: IBM Watsonx Code Assistant for Red Hat Ansible Lightspeed. Source: [10]

In the context of computer networking, Ansible Lightspeed is particularly valuable for automating both routine and complex tasks. These include network device provisioning, firmware upgrades, and policy enforcement to meet compliance requirements. By simplifying and standardizing these processes, Ansible Lightspeed reduces operational overhead, enhances consistency, and enables faster response times. These benefits are especially important in dynamic network environments, where even minor misconfigurations or delays can lead to significant disruptions. Figure 2.8 demonstrates how IBM Watsonx Code Assistant integrates into the Ansible Lightspeed workflow, effectively translating human intent into executable automation code and bridging the gap between natural language and infrastructure management.

Chapter 3

AI Techniques for Network Creation

Artificial Intelligence is changing the way networks are designed and built. Traditional approaches to network creation often rely on manual planning and fixed rules, which can struggle to keep up with today's increasingly complex and large-scale infrastructures. In contrast, AI offers smarter, more flexible methods, such as evolutionary algorithms that can optimize network topology and reinforcement learning agents that adaptively configure networks based on changing conditions.

In this chapter, we explore how AI is being applied to the creation and planning of computer networks. Specifically, we review and summarize three research papers that demonstrate the use of different AI techniques in this area. Each study highlights a unique approach to solving common network design challenges using intelligent, data-driven strategies.

3.1 Network Topology Optimization

Designing an efficient network topology is one of the key challenges in building and operating modern computer networks. The goal is to determine the best arrangement of nodes and connections to meet specific performance targets, such as reducing latency, maximizing throughput, improving fault tolerance, and keeping operational costs low. Traditionally, this has been done using heuristic methods and manual tuning. While these approaches can work, they are often time-consuming, error-prone, and may fall short of delivering optimal results, especially in large-scale or rapidly changing network environments.

AI offers a new and powerful alternative. By leveraging AI, network topology optimization can become more intelligent, automated, and adaptable. AI algorithms can process large volumes of network data, learn from past performance, and make informed decisions about the best topological configurations. Techniques such as Genetic Algorithms (GA), RL, and Graph Neural Networks (GNNs) are increasingly being used to address this problem. These methods can adapt in real time to shifting network demands, balance multiple performance goals, and help create topologies that are both efficient and resilient, an important advantage for today's networks, which support technologies like Cloud Computing, IOT, and 5G networks.

In the following two subsections, we'll look at how two different AI approaches, RL and GA, have been applied to the problem of network topology optimization, based on insights from recent research papers.

3.1.1 Reinforcement Learning (RL):

RL is especially well-suited for complex, dynamic environments like computer networks, where conditions can change frequently due to fluctuating traffic loads, hardware failures, or shifting user demands. Instead of relying on fixed rules or exhaustive manual configurations, RL enables a system to learn from experience. Through a process of trial and error, a RL agent makes decisions, receives feedback in the form of performance metrics (like latency or throughput), and gradually learns strategies that lead to better outcomes. This makes RL a powerful tool for designing network topologies

that can adapt in real time to varying operational needs.

A compelling application of this approach is presented in a 2023 study by Zhuoran Li et al. [13], where the authors address the highly complex problem of network topology optimization. As networks grow in size and functionality, identifying the best way to connect nodes becomes increasingly challenging. Traditional heuristic-based methods often fall short in this area because they can not easily handle the enormous number of possible configurations or the operational constraints that real networks must satisfy.

To tackle these issues, the authors propose a novel DRL framework called Advantage Actor Critic-Graph Searching (A2C-GS). This framework is designed to explore large network topology spaces efficiently while balancing performance improvements with the cost and feasibility of reconfigurations. The A2C-GS method is composed of three main components:

- **Representation Component:** This part reduces the complexity of the network's state and action spaces by analyzing the objective function. By simplifying the learning environment, the agent can focus on the most relevant factors, making the training process more efficient.
- **DRL Actor Layer:** This layer is responsible for exploring different topology configurations. Instead of randomly trying out every possibility, it uses guided exploration to find the most promising designs based on accumulated learning.
- **Verifier:** Once a potential topology is generated, the verifier ensures that it is not only high performing but also feasible. It checks for things like valid routing paths, structural constraints, and compliance with performance requirements.

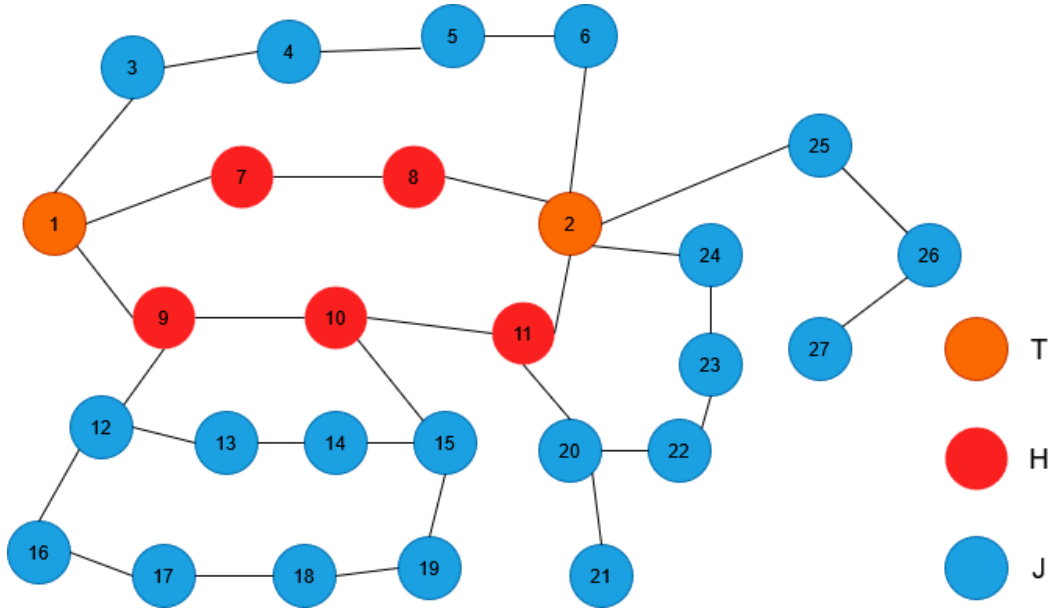


Figure 3.1: An example of a network topology. Source: [13]

This study demonstrates that A2C-GS can outperform traditional heuristic methods in both solution quality and computational efficiency. In a real-world case study, the framework successfully identified high-performing network topologies that enhanced key metrics such as link utilization and overall throughput, while maintaining low latency. What makes this approach particularly compelling is its scalability, which effectively manages large, complex network environments, and its adaptability, continuously learning to improve performance without manual tuning. By combining DRL with graph-based network representations, this work highlights a promising direction for building more intelligent, automated, and scalable solutions to network topology optimization.

3.1.2 Genetic Algorithms (GAs)

GA are a class of optimization techniques inspired by the principles of natural evolution. Just like biological systems evolve to become more efficient and resilient, GA algorithms are designed to evolve solutions to complex problems by mimicking processes such as selection, crossover, and mutation. In the context of network design, GA are particularly useful for navigating large and complex solution spaces, where the number of possible topology configurations makes brute-force or manual approaches impractical.

GA are especially powerful in tackling **multi-objective optimization problems**, which are common in network topology design. These problems often require balancing several competing goals, such as minimizing infrastructure cost, maximizing performance, ensuring reliability, and maintaining QOS. Because of their ability to explore a diverse set of potential solutions and converge toward optimal or near-optimal results, GA are well-suited for modern network environments that are large-scale, dynamic, and increasingly layered.

A notable example of this approach is presented in the study by Uwe Bauknecht titled "A Genetic Algorithm Approach to Virtual Topology Design for Multi-Layer Communication Networks" [5]. In this work, Bauknecht introduces a GA-based method aimed at optimizing virtual topologies across multiple network layers. The method is designed for scenarios where traffic patterns, physical constraints, and logical connectivity must all be considered simultaneously, common in large-scale data centers and carrier-grade networks.

In the proposed approach, candidate topologies are represented as chromosomes, with each gene encoding a logical link between core routers in the network. The GA operates by iteratively improving these chromosomes over multiple generations:

- **Selection** chooses the most promising topologies based on a fitness function that accounts for cost efficiency and QOS compliance.
- **Crossover** combines elements from two high-performing topologies to generate new candidate solutions.
- **Mutation** introduces slight random variations to maintain diversity in the population and avoid premature convergence.

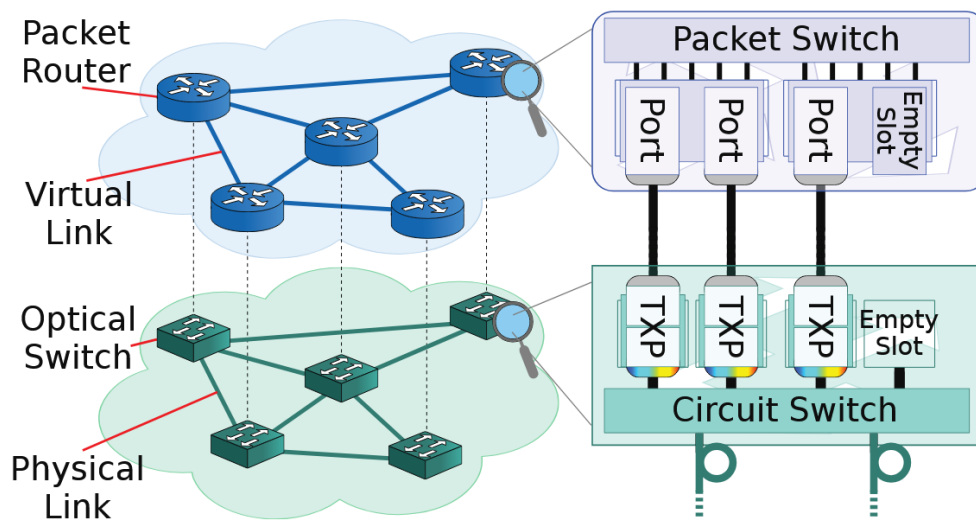


Figure 3.2: An example of a multi-layer network. Source: [5]

The study's results show that the GA-based approach significantly outperforms traditional heuristic methods, especially in large and complex network topologies where exhaustive search is computationally infeasible. The algorithm was able to

reduce resource consumption and infrastructure cost while still meeting QOS requirements, demonstrating its effectiveness for real-world network design tasks. Importantly, this research highlights how evolutionary algorithms like GA offer a scalable and adaptive solution for network topology optimization. As modern networks continue to evolve, driven by trends such as virtualization, cloud computing, and SDN, the ability to automatically generate and refine efficient topologies becomes increasingly valuable.

Bauknecht's work provides strong evidence for the practical value of GA in network creation. These techniques allow designers to explore complex solution spaces more effectively than traditional methods, leading to better performance, cost-efficiency, and flexibility in today's rapidly evolving networking landscape.

3.2 Network Planing

Network planning is a critical step in the design and operation of modern communication systems. It involves making decisions about how to structure the network, determining where to place links and devices, how to allocate capacity, how to route traffic, and how to ensure the system can handle failures or demand fluctuations. The goal is to meet specific performance targets, like low latency and high throughput, while also keeping costs and resource usage under control.

Traditionally, network planning problems have been solved using mathematical models such as Integer Linear Programming (ILP). However, as networks grow in size and complexity, with the rise of Cloud Computing, data centers, and distributed architectures, ILP-based methods can become computationally impractical. They often require significant processing time and rely on fixed assumptions that don't easily adapt to real-world variability, such as changing traffic patterns or hardware failures.

Network Planning with Deep Reinforcement Learning

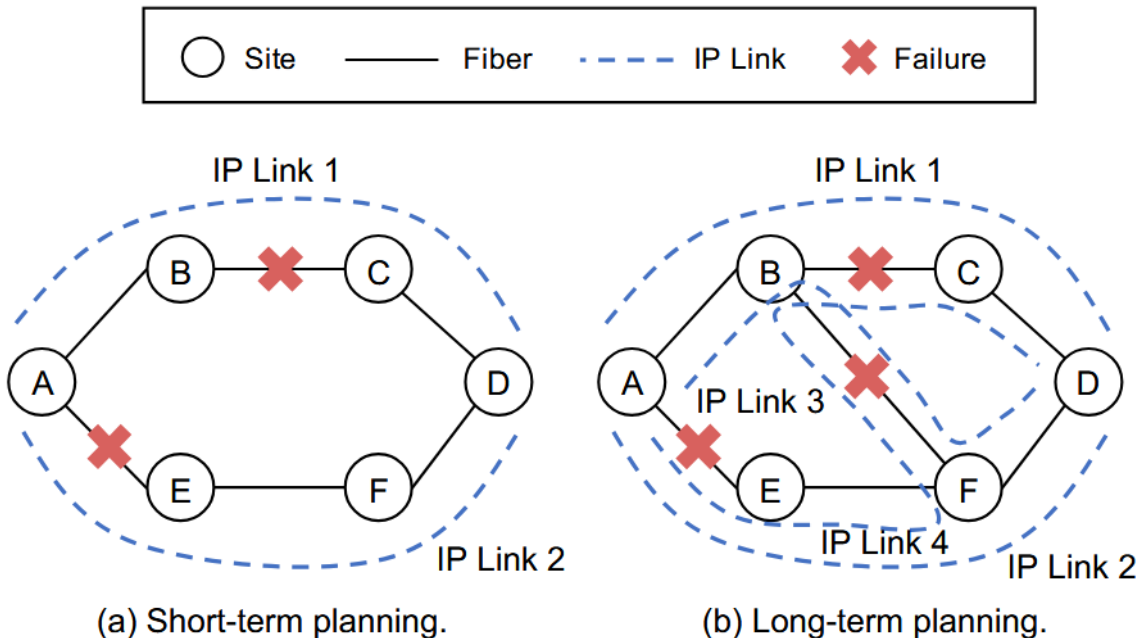


Figure 3.3: An example for network planning to satisfy a 100Gbps flow from A to D under any of the three single-fiber failures. Source: [24]

To overcome these limitations, researchers have started exploring more flexible and scalable approaches that integrate AI

with traditional optimization techniques. One notable example is **NeuroPlan**, a method introduced by Hang Zhu and colleagues in 2021 [24]. This approach combines RL and graph-based modeling with classical ILP to make the network planning process more efficient and adaptable. **NeuroPlan** operates in two main stages:

1. **Exploration with Reinforcement Learning:** In the first stage, a reinforcement learning agent interacts with a model of the network. Through trial and error, it learns which configurations are most promising in terms of performance and cost. This process helps narrow down the vast number of possible network designs, reducing the search space significantly. Rather than exploring every possible topology, the agent focuses only on those that are likely to yield good results.
2. **Refinement with ILP:** Once a smaller set of candidate configurations is identified, the second stage uses ILP to fine-tune the final solution. This ensures that all technical constraints are satisfied, such as link capacities, redundancy requirements, and routing limits, while still optimizing for objectives like cost efficiency and robustness.

A key innovation in **NeuroPlan** is how it represents network structures. Because computer networks are inherently graph-based, with nodes and links forming complex interconnections, the authors use GNNs to help the model better understand these relationships. GNNs allow the RL agent to learn from the structural properties of the network, improving its ability to make planning decisions that take into account the overall topology and dependencies between components.

When tested on real-world network datasets, **NeuroPlan** was shown to reduce planning costs by up to 17% compared to manually tuned baseline methods. It also performed well in large-scale scenarios, where traditional planning approaches often struggle due to the exponential growth in possible configurations. This makes **NeuroPlan** particularly promising for next-generation network infrastructures, where scale and adaptability are crucial.

NeuroPlan illustrates the power of combining advanced AI techniques with classical optimization tools. By blending RL, graph-based modeling, and ILP, it provides a more intelligent and scalable approach to network planning. As networks become more complex and dynamic, hybrid solutions like this will likely play a key role in enabling efficient, resilient, and cost-effective communication systems.

Chapter 4

AI Techniques for Network Administration

This chapter explores the application of key AI techniques in network administration, with a particular focus on network traffic prediction and predictive maintenance. It begins by examining two distinct AI approaches for addressing network traffic prediction. The discussion then shifts to how predictive maintenance can be leveraged to improve network performance, reduce downtime, and support the development of smarter, more adaptive, and self-managing network systems.

4.1 Network Traffic Prediction

Predicting network traffic accurately is more important than ever in today's increasingly complex digital landscape. As more organizations rely on Cloud services, IOT devices, 5G networks, and Edge Computing, managing traffic efficiently has become essential for keeping networks running smoothly. Being able to anticipate how traffic will behave helps improve overall performance, avoid congestion, allocate resources more intelligently, and maintain a consistent level of service.

Traditional traffic prediction methods, like rule-based systems or statistical models, often struggle to keep up with the fast-changing and unpredictable nature of modern networks. These older approaches were not designed to handle the high variability, non-linear trends, and massive scale that define today's network environments. To address these limitations, AI techniques, and in particular ML and DL, have emerged as a promising solution. AI-based traffic prediction uses historical network data to recognize patterns and forecast future traffic behavior. Depending on the nature of the data and what kind of predictions are needed, different AI techniques can be applied.

4.1.1 Machine Learning Approaches

ML techniques are widely used in traffic prediction due to their ability to model complex relationships within structured datasets. Algorithms like SVM, Random Forests, and ensemble methods such as Bagging and Gradient Boosting have been applied to tasks including predicting traffic volume, estimating packet arrival times, and classifying network flow types. These models tend to perform best when the data is well-labeled and the input features are clearly defined.

One of the key advantages of ML models is their efficiency, they are generally faster to train and require fewer computing resources compared to deep learning models. This makes them suitable for real-time or resource-constrained environments. However, ML models do have limitations. Most notably, they depend on manual feature engineering, which involves selecting relevant input features based on expert knowledge. This process can be both time-consuming and prone to oversight. Additionally, many traditional ML models are not well-equipped to capture temporal dependencies, which are critical for modeling traffic that varies over time.

A practical example of ML applied to real-world network traffic prediction is found in the work by Alekseeva et al. (2021) [2]. In their study, the authors compared several classical ML models, including Linear Regression, Gradient Boosting, Random Forest, and SVM, using a dataset from a real LTE wireless network. They found that Gradient Boosting achieved

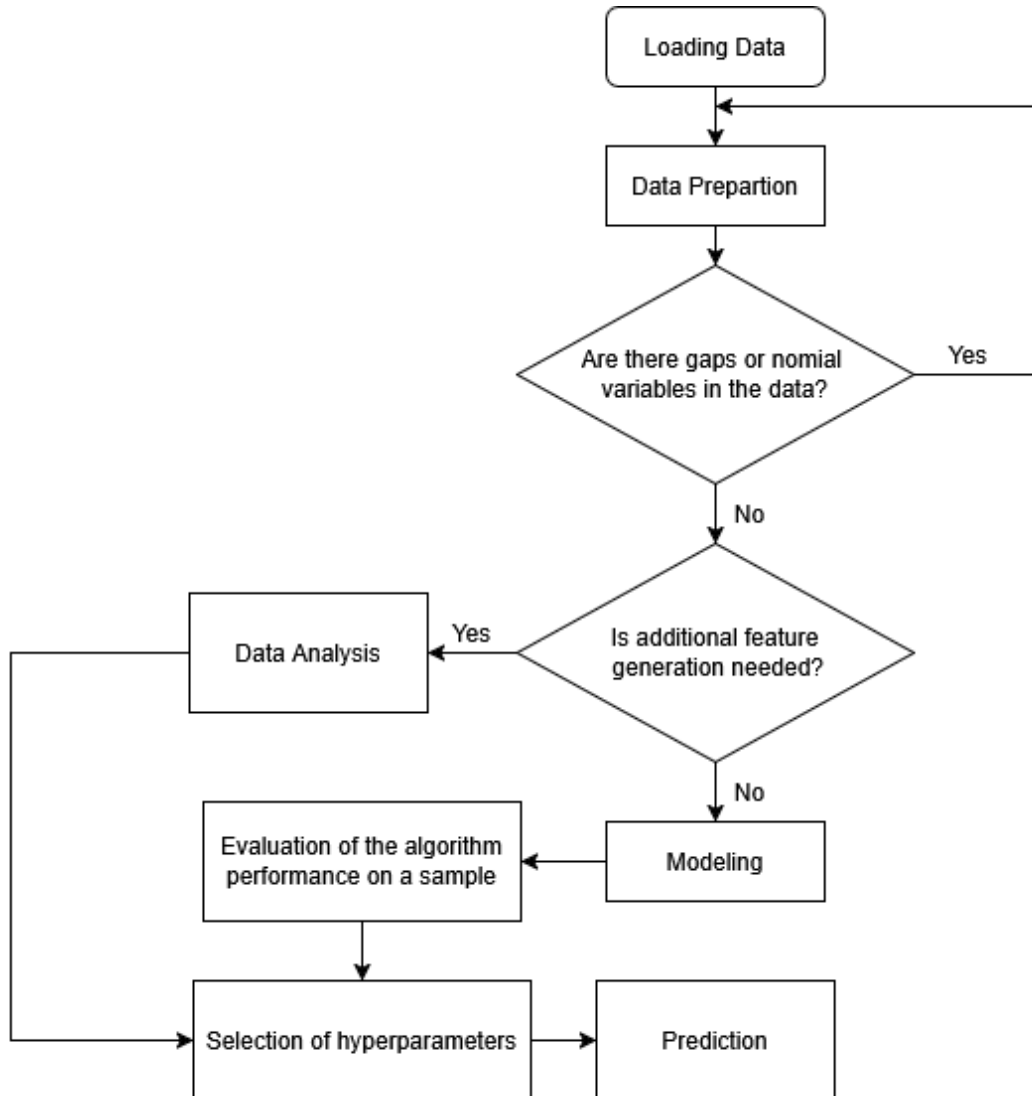


Figure 4.1: Algorithm for creating a traffic predicting model. Source: [2]

the highest prediction accuracy, while SVM was the fastest to train. Interestingly, Random Forest performed the worst in this case, possibly due to its handling of feature representations in the dataset. This study emphasizes that the effectiveness of a given model often depends on the specific performance needs of the task, whether that's accuracy, speed, or ease of deployment.

Despite their limitations, ML models remain a reliable and accessible choice for many network traffic prediction tasks. They are especially useful in situations where traffic patterns are relatively stable and fast, and interpretable outputs are needed. When applied appropriately, ML techniques can help network operators make more proactive and informed decisions, reducing the risk of congestion and enhancing overall network reliability.

4.1.2 Deep Learning Approaches

DL has brought a major shift in how we approach network traffic prediction, offering tools that can learn directly from raw or lightly processed data. This allows models to handle the complex, fast-changing, and high-dimensional nature of today's networks far more effectively than traditional machine learning methods, which often depend on manual feature

selection and have trouble capturing long-term patterns.

One of the most commonly used deep learning architectures for traffic forecasting is the RNNS, especially more advanced variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs). These models are designed to work with sequential data, which makes them ideal for analyzing time-series traffic patterns. LSTM, in particular, excels at identifying long-term dependencies, like recurring daily peaks or delayed impacts of network events, that simpler models may miss. In the study “Network Traffic Prediction Using Recurrent Neural Networks” [16], Ramakrishnan and Soni applied RNNS, LSTM, and GRUs to real-world datasets from GEANT and Abilene (two major internet backbone networks). Their findings showed that LSTM and GRUs models not only outperformed traditional techniques in prediction accuracy but also demonstrated strong adaptability in handling fluctuating and diverse traffic types.

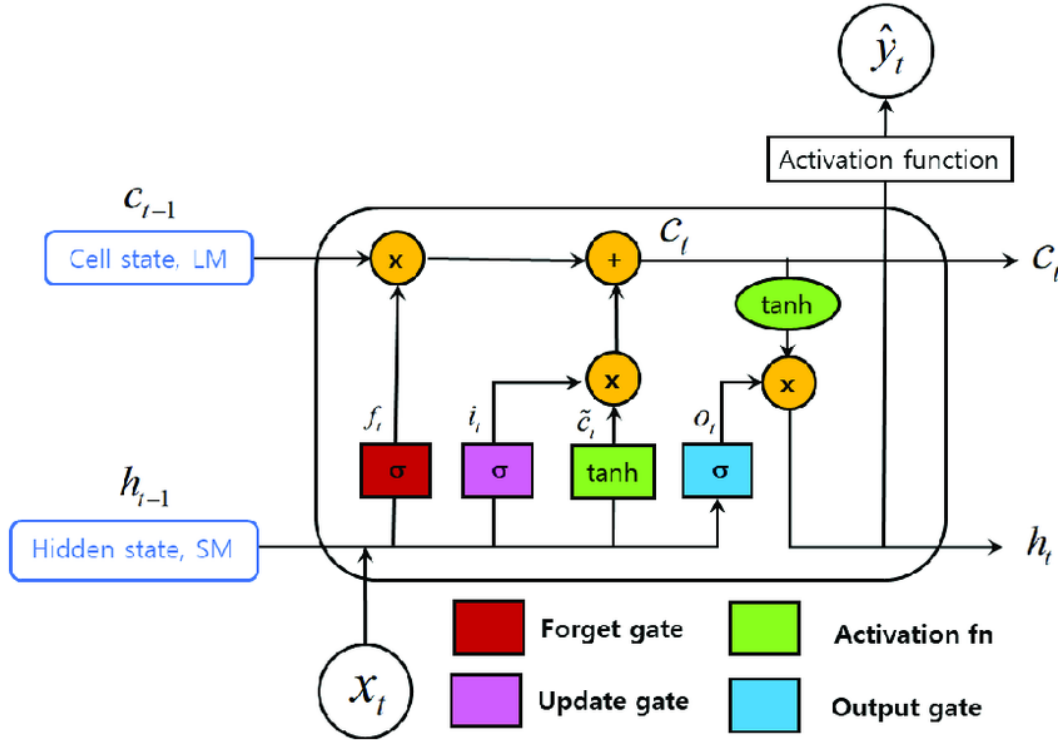


Figure 4.2: Diagram of LSTM cell structure. Source: [14]

Another emerging and increasingly relevant technique in this space is the use of GNNs. Because computer networks are naturally structured as graphs, with nodes representing devices or routers and edges representing communication links, GNNs are uniquely suited to model both the structure and flow of traffic across a network. These models go beyond sequential learning by capturing spatial dependencies between network elements, allowing for more context-aware predictions. For example, GNNs can help identify how congestion in one part of the network might affect neighboring segments, which is critical for dynamic routing and load balancing.

The recent survey “Graph Neural Networks for Intelligent Modelling in Network Management and Orchestration” by Tam. et al [20] explores how GNNs are being used to enhance automation and decision-making in network operations. It highlights applications across several areas, including traffic prediction, fault detection, and resource optimization. The authors note that GNNs provide a powerful way to integrate both topological and temporal information, making them a strong candidate for the future of intelligent, self-managing networks.

Overall, DL approaches offer powerful capabilities for traffic prediction, particularly in large-scale, complex environments. While these models require significant computational resources and substantial amounts of data to train, their ability to uncover subtle patterns and adapt to dynamic conditions makes them increasingly essential in modern network management. As networks continue to evolve in scale and complexity, DL will play a key role in building smarter, more efficient, and more reliable infrastructure.

4.2 Predictive Maintenance

Maintaining the health and stability of network infrastructure is essential for ensuring uninterrupted services, especially as networks grow in size and complexity. Traditional approaches to maintenance, such as reactive strategies that respond only after faults occur, or scheduled preventive maintenance that doesn't consider actual equipment condition, are no longer sufficient in dynamic, high-demand environments. These methods often lead to unnecessary downtime, wasted resources, or even catastrophic failures if issues go unnoticed.

AI offers a powerful alternative through predictive maintenance, which focuses on anticipating and preventing problems before they occur. By leveraging ML and DL techniques, AI systems can monitor network components in real time, learn from historical data, and identify patterns that precede hardware or performance failures. This allows administrators to take timely, informed actions that improve reliability and reduce maintenance costs.

AI-based predictive maintenance systems typically follow a structured workflow. First, data is continuously collected from network devices such as routers, switches, and servers. This data may include metrics like CPU usage, packet loss, latency, temperature, and system logs. The collected data is then preprocessed and analyzed by AI models, which are trained to recognize early indicators of degradation or failure. When the system detects unusual behavior that suggests a potential issue, it can trigger alerts or even initiate automated corrective actions, such as rerouting traffic or scheduling a maintenance task.

The paper "Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends" by Ucar et al. [21] provides a thorough examination of how AI is transforming maintenance strategies across industries, including networking. The authors highlight several key components essential to building effective predictive maintenance systems, including accurate data pipelines, robust AI models, intuitive decision-making modules, and clear user interfaces. A particularly important theme in their study is the issue of trustworthiness in AI. The paper discusses the need for explainable and transparent models, especially in critical systems like computer networks, where operators must be able to understand and verify AI-generated recommendations.

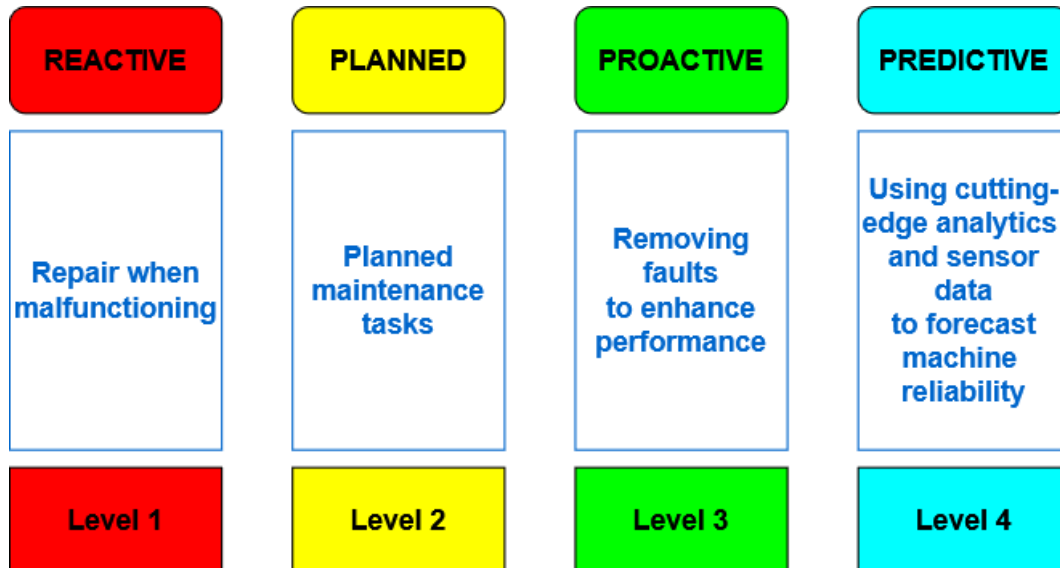


Figure 4.3: Different levels of system maintenance. Source: [21]

The study also outlines future directions for the field, such as integrating AI with digital twins (virtual models of physical systems), leveraging generative AI for automated decision support, and applying predictive analytics within Industrial IOT environments. These trends point toward more adaptive, autonomous systems capable of maintaining themselves with minimal human intervention.

When applied to computer networks, AI-powered predictive maintenance offers a wide range of benefits. It enhances network reliability by identifying and resolving problems before they impact users. It improves operational efficiency

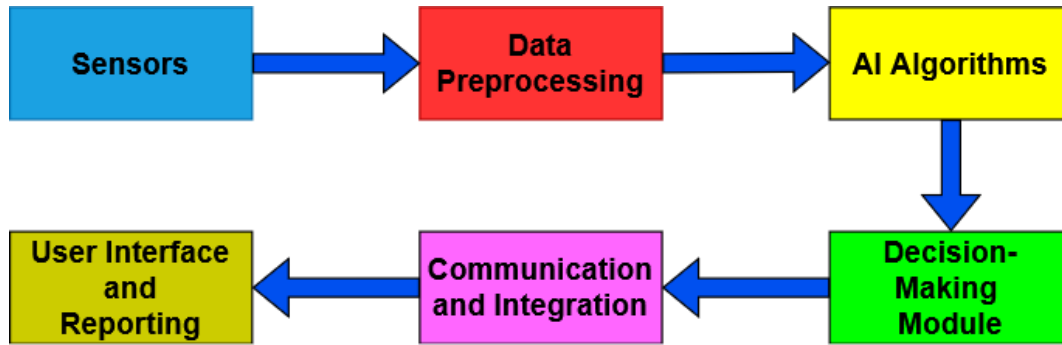


Figure 4.4: Key components of an AI-based Predictive Maintenance System. Source: [21]

by targeting maintenance efforts only where needed. It also reduces downtime, cuts operational costs, and supports more strategic resource allocation. Predictive maintenance represents a major step forward in network administration. By shifting from reactive problem-solving to proactive planning, AI allows network operators to stay ahead of potential disruptions and keep infrastructure running smoothly, even in the face of increasing scale and complexity. As networks continue to evolve, integrating predictive maintenance powered by AI will be a crucial component of building resilient, intelligent communication systems.

Chapter 5

AI Techniques for Network Security

One of the most widely adopted applications of AI in the field of computer networking is in network security. As cyber threats become more sophisticated and networks grow in complexity, traditional security measures often fall short in detecting and responding to emerging threats in real time. AI offers a powerful alternative by enabling more intelligent, adaptive, and proactive approaches to securing network infrastructures.

In this chapter, we explore how AI is being used to enhance network security, with a particular focus on two critical areas: anomaly detection and threat detection. These techniques allow systems to recognize malicious activity, such as intrusions, zero-day attacks, and unusual behavior patterns, before significant damage occurs, often without relying on predefined rules or signatures.

We also examine the ethical considerations and challenges that come with integrating AI into security systems. While AI can significantly improve response times and detection accuracy, it also raises important questions around transparency, fairness, accountability, and the potential for unintended consequences. By examining both the capabilities and the ethical dimensions of AI in network security, this chapter provides a balanced perspective on how these technologies are reshaping the future of cybersecurity.

5.1 Threat Detection and Anomaly Detection with AI

In today's increasingly connected and complex network environments, security is a growing concern. The rapid evolution and growing sophistication of cyber threats have exposed the limitations of traditional, rule-based security systems. These conventional approaches typically depend on predefined signatures or manually crafted rules, making them ill-suited for detecting novel, unknown, or highly sophisticated attacks. To address these challenges, AI, particularly ML and DL, has emerged as a powerful and adaptive solution, especially in the areas of threat detection and anomaly detection.

AI-driven threat detection harnesses the power of learning algorithms to identify patterns across massive volumes of network data. Unlike traditional systems that only flag known threats, AI models can analyze behaviors and spot unusual activity that might indicate a security breach, such as data exfiltration, Distributed Denial of Service (DDoS) attacks, or insider threats. These techniques are typically grouped into two main categories: anomaly detection and threat detection.

Anomaly Detection focuses on identifying deviations from established norms. AI models, especially Unsupervised or Semi-supervised Learning algorithms, monitor traffic patterns, system logs, and user behavior to detect anything unusual. These models can build a baseline of "normal" activity and flag any deviations in real-time. Algorithms such as K-means Clustering, Autoencoders, and Isolation Forests are commonly used in this domain. Because they don't rely on labeled datasets, these models are particularly valuable for identifying previously unseen attack methods or zero-day threats.

Threat Detection, on the other hand, often uses Supervised Learning. These models are trained on datasets labeled with known threat types and can classify incoming network activity as benign or malicious. Techniques like SVM, Decision Trees, and increasingly, DL architectures like CNNs and RNNs, have proven effective in identifying patterns consistent with malware, phishing, or brute-force attacks.

Graph-based approaches are also gaining popularity. GNNs are particularly well-suited for cybersecurity tasks, as network environments can naturally be represented as graphs, with devices as nodes and communications as edges. GNNs can model relationships between users, devices, and activities over time, making them powerful tools for detecting lateral movement and coordinated attacks within a network.

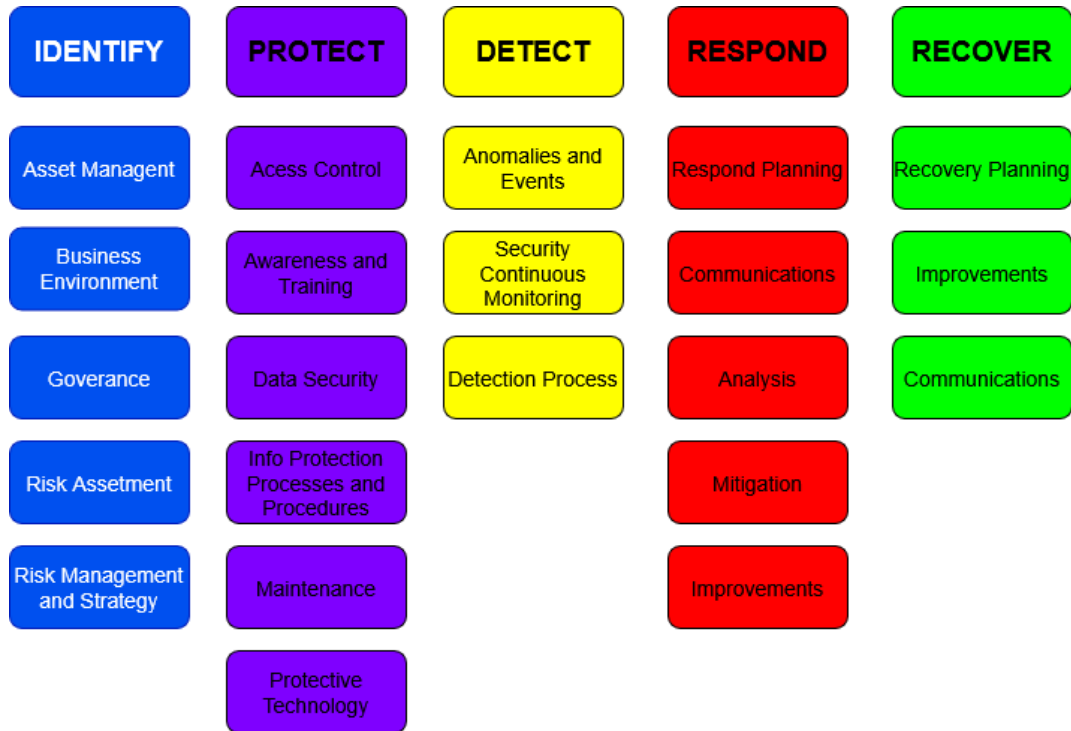


Figure 5.1: NIST cybersecurity framework. Source: [11]

To better understand the state of research in this space, the study by Kaur et al. (2023) [11], titled “Artificial Intelligence for Cybersecurity: Literature Review and Future Research Direction”, provides an extensive overview of AI’s role in cybersecurity. The authors reviewed 2,395 studies and analyzed 236 in detail, organizing their findings using the National Institute of Standards and Technology (NIST) cybersecurity framework: Identify, Protect, Detect, Respond, and Recover. This structure gives a comprehensive look at how AI is being used at every stage of the cybersecurity lifecycle.

One of the most valuable contributions of this review is its detailed taxonomy of AI applications across different security functions. It outlines how AI is used for malware detection, phishing detection, Intrusion Detection Systems (IDS), and automated response mechanisms. It also identifies a growing focus on Explainable AI (XAI), an area where models are designed not just to make predictions, but to provide understandable reasons for their decisions, which is essential in high-stakes cybersecurity contexts.

The paper also highlights some important gaps in current research. For example, there is relatively little work focusing on AI’s role in recovery after a cyberattack, a function that is essential for building resilient systems. The authors also stress the need for more trustworthy, ethical, and transparent AI systems that can be reliably integrated into existing security frameworks.

Complementing this perspective, the study by Kashif Naseer Qureshi, titled “Anomaly Detection and Trust Authority in Artificial Intelligence and Cloud Computing” [15], further explores the challenges of integrating AI into cybersecurity, particularly within cloud environments. Qureshi’s work focuses on the intersection of anomaly detection and trust management, proposing models that not only identify unusual behavior but also assess the trustworthiness of users and systems. This dual approach enhances overall security by combining behavioral analysis with dynamic trust evaluation.

Qureshi emphasizes the need for intelligent systems that can adapt to evolving threats while maintaining a high level of trust among users and services in distributed, cloud-based architectures. The study also underlines the importance of

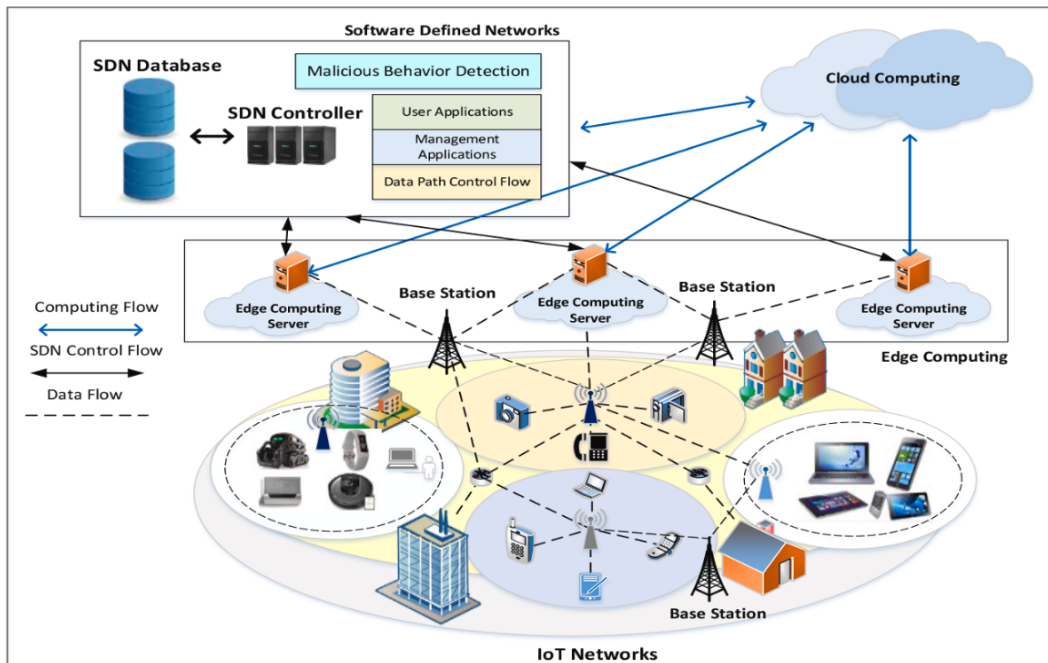


Figure 5.2: SDN-based Edge Computing architecture for IOT. Source: [15]

decentralized trust authorities and outlines mechanisms to improve reliability, scalability, and transparency—issues that are increasingly important as cloud computing becomes more ubiquitous.

Together, these studies illustrate both the breadth and depth of AI applications in cybersecurity, highlighting promising directions such as explainability, resilience, trust management, and recovery, areas that will be critical for the next generation of secure, AI-driven network systems.

5.2 Ethical Implications and Challenges in AI-Driven Security

Despite its strengths, AI-driven security is not without its challenges. One of the most pressing issues is the potential for false positives, situations where normal activity is mistakenly flagged as malicious. These can overwhelm security teams with unnecessary alerts, reducing overall effectiveness. Additionally, AI systems can be vulnerable to adversarial attacks, where malicious actors intentionally manipulate inputs to deceive the model. This raises serious concerns about the reliability and robustness of AI in high-stakes environments.

Another significant challenge lies in the complexity and opacity of many AI models, particularly DL systems. These models often function as "black boxes," making it difficult for human analysts to understand why a particular alert was triggered or how a specific decision was made. This lack of interpretability can hinder timely and informed responses to threats and erode trust in automated security solutions.

The quality, diversity, and representativeness of training data also play a critical role in the performance of AI systems. If the data used to train models is biased, outdated, or incomplete, the system may fail to detect certain types of threats or, worse, reinforce existing biases. This can lead to uneven protection across different environments and increase the risk of missed attacks.

Nevertheless, AI offers clear advantages over traditional methods, especially in terms of scalability, adaptability, and the ability to detect previously unknown threats. As networks continue to evolve and threats grow more sophisticated, AI will play an increasingly vital role in securing digital infrastructures. Continued research, as emphasized by Kaur et al., will be essential to address existing gaps and ensure that AI technologies remain robust, transparent, and aligned with broader cybersecurity goals.

Chapter 6

AI techniques Implementation

While the previous chapters explored the theoretical foundations and academic research surrounding the use of AI in computer networking, this chapter shifts focus to a practical implementation. By putting theory into practice, this implementation helps to bridge the gap between academic concepts and real-world applications. It explores how AI can be used to enhance network performance and adaptability, particularly in dynamic scenarios where manual configuration would be time-consuming or inefficient.

The following sections outline the project's architecture, setup, and results, highlighting key takeaways and challenges encountered along the way. This hands-on approach provides a clearer understanding of what it takes to integrate AI into modern networking systems and the advantages that such integration can offer.

6.1 Methodology and Development Environment Setup

Before diving into the technical implementation, it is important to outline the approach and development environment used throughout this chapter. Due to limited experience with configuring physical network hardware such as routers and switches, all experiments in this study were conducted using software-based simulations. This approach not only made testing more accessible but also ensures that others can more easily replicate the work on their systems.

Python 3 was selected as the primary programming language, owing to its widespread use in the fields of ML and DL, and its extensive ecosystem of libraries and tools. Each project was developed within a dedicated Python 3 virtual environment, allowing dependencies and packages to be managed in isolation. In some cases, older versions of certain libraries were intentionally used, as specific features required by the implementation had been deprecated or removed in newer releases. As such, maintaining an environment consistent with the documented configurations is essential; small differences in versions or system settings can cause errors or unexpected behavior.

All experiments were performed on a Lenovo ThinkPad E14 Gen 4 running Ubuntu 24.04 LTS. The choice of a Linux-based operating system was based both on the author's familiarity with Linux and on practical considerations, as many of the tools and libraries used are either optimized for or exclusive to Linux environments. Consequently, the code may not run as intended on Windows or macOS without additional adjustments.

This chapter aims to offer a practical, hands-on example of how AI techniques can be applied to networking challenges. While the implementations are not designed for production use, they serve as proof-of-concept demonstrations of the ideas discussed in earlier chapters. The goal is to highlight the feasibility of these techniques and encourage further exploration and experimentation in the field of AI-driven computer networking.

6.2 SDN Topology Optimization Using Reinforcement Learning

To demonstrate how AI techniques can be applied to network creation, this section presents a project that integrates a RL agent with a SDN environment. The primary objective is to optimize network topology dynamically, with the goal of minimizing average latency across the network.

The project architecture consists of four key components:

- **Mininet** – used to emulate a flexible and customizable virtual network topology.
- **OpenDaylight** – serving as the SDN controller, responsible for managing the network via the OpenFlow protocol. It also provides a graphical interface through the DLUX web interface to visualize and monitor the network topology.
- **TensorFlow Agents** – providing the reinforcement learning framework used to train and manage the agent.
- **A Deep Q-Network (DQN) agent** – trained to observe network metrics and learn optimal topology control strategies that improve performance over time.

This implementation offers a practical, hands-on demonstration of how RL can enable automated, data-driven decision-making in SDN.

6.2.1 Project Workflow

The basic workflow of the project is outlined below:

1. **Topology Initialization:** An initial network topology is created using Mininet and brought under the control of the OpenDaylight (ODL) SDN controller. This forms the foundational environment for training and evaluating the RL agent.
2. **Training Phase:** In this phase, the RL agent interacts with the SDN environment through a series of training episodes. At each step, the agent selects actions intended to reduce overall end-to-end network latency. Based on the results of these actions, the agent updates its policy to improve future decision-making.
3. **Policy Deployment:** After training is complete, the optimized policy derived by the RL agent is applied to the network in real time. This modifies routing decisions or link weights based on the learned strategy.
4. **Performance Evaluation:** The optimized policy's effectiveness is measured by conducting latency tests using Mininet's `pingall` command before and after policy deployment. The results are recorded in a log file (`result.txt`). Additionally, screenshots of the network topology, taken via the OpenDaylight DLUX interface, are used to visually compare the network state before and after training.

6.2.2 Deployment and Execution Instructions

The project source is available on GitHub at: <https://github.com/NagiLam/FrankfurtUAS/tree/main/Thesis>
Look for the file named `rl_sdn_topology.zip`, which contains all necessary code and resources.

Step 1: Setting Up OpenDaylight

For this project, it is essential to use OpenDaylight version 0.8.4, as the DLUX web interface module has been deprecated in more recent versions and may not function as expected. Ensuring compatibility between software components is critical, particularly for core dependencies such as TensorFlow and TF-Agents, as newer versions may introduce breaking changes that affect the stability or correctness of the implementation.

Step 3: Accessing the DLUX Web Interface

To view the network topology, open a web browser and navigate to:

`http://localhost:8181/index.html#/login`

Use the following default credentials to log in:

- **Username:** admin
- **Password:** admin

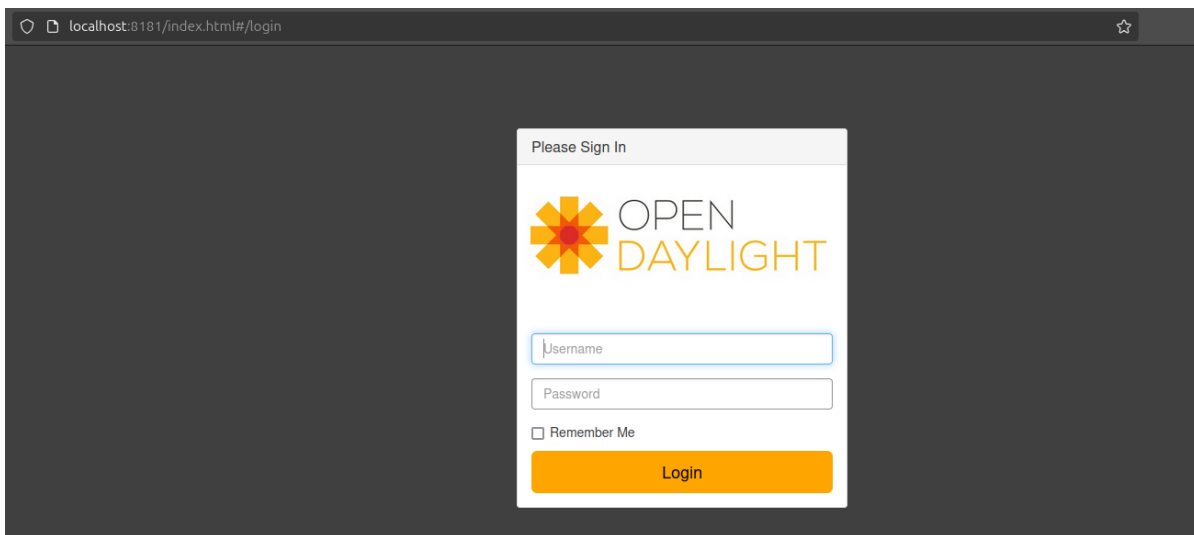


Figure 6.2: OpenDaylight DLUX login page accessible via the web interface.

Step 4: Setting Up the Python Virtual Environment

The project is built using Python 3.11 and relies on specific versions of TensorFlow (2.15) and TF-Agents (0.19). It is recommended to use a Python virtual environment to avoid version conflicts.

To set up the virtual environment, run the following commands:

```
python3.11 -m venv tensorflow_311
source tensorflow_311/bin/activate
pip install --upgrade pip
pip install tensorflow==2.15.0
pip install tf-agents==0.19.0
pip install numpy==1.26.4
pip install matplotlib mininet absl-py gym requests
```

Ensure that all required Python packages are installed without errors before proceeding.

Step 5: Running the Project

After the OpenDaylight controller is running and the Python environment is ready, unzip the project file `rl_sdn_topology.zip` and navigate into the extracted directory:

```
unzip rl_sdn_topology.zip
cd rl_sdn_topology
```

To run the entire process, execute the combined evaluation and training script:

```
sudo python3 eval_latency.py
```

This script performs the following tasks:

- Initializes a base network topology in Mininet.
- Measures baseline latency using `pingall`.
- Trains a RL agent to improve network topology.
- Applies the learned policy to modify the topology.
- Measures latency again to verify improvement.
- Captures the topology structure before and after training via OpenDaylight.
- Saves both latency values and topology images for comparison.

Step 6: Viewing Results

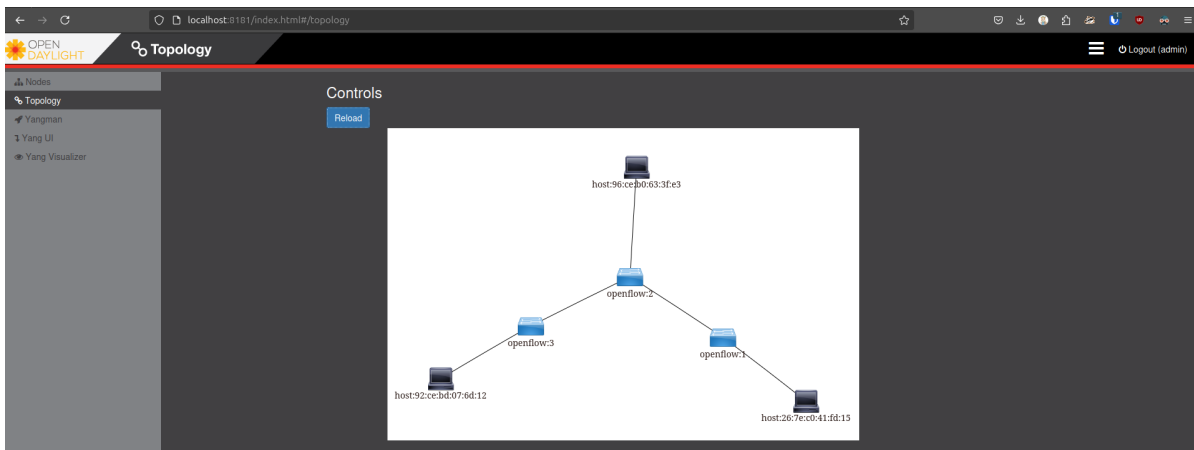


Figure 6.3: Visualization of the network topology displayed in the OpenDaylight DLUX web interface.

After the script completes, two types of output are generated:

- `result.txt` – contains latency measurements before and after applying the trained policy.
- `topology_comparison.png` – a side-by-side image of the network topology before and after training, as visualized in the OpenDaylight DLUX interface (Figure 6.4).

These results demonstrate the effectiveness of the RL approach in optimizing network topology to reduce communication latency. By comparing both the network structure and performance metrics before and after the training phase, the project highlights the capability of RL to assist in automated decision-making within SDN environments. The trained agent successfully identified and applied topology adjustments that resulted in lower end-to-end latency, thereby improving overall network efficiency under simulated traffic conditions. Quantitative evaluation supports this improvement: before training, latency values were recorded at $[1537.574, 0.279, 0.247]$ milliseconds, whereas after applying the learned policy, they decreased significantly to $[0.159, 0.148, 0.176]$ milliseconds. These findings confirm that the RL-based strategy was able to make targeted and beneficial modifications to the network configuration, leading to measurable performance gains.

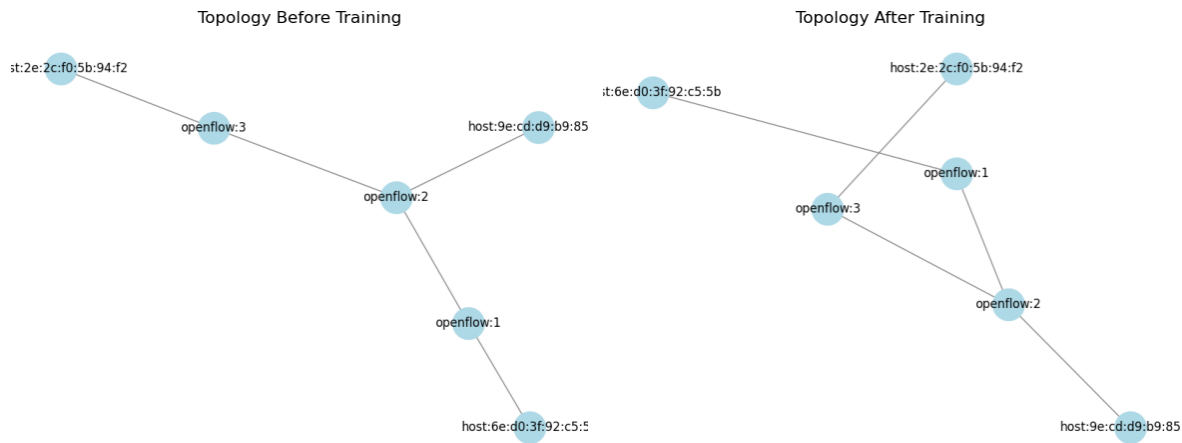


Figure 6.4: The network topology before and after training.

6.3 Smart Network Traffic Classification Using Machine Learning

To demonstrate how AI techniques can be applied to network administration, this section presents the implementation of a smart traffic classification system based on supervised ML. The project automates the end-to-end process of capturing, extracting, and classifying network traffic using statistical features derived from packet-level data. The primary objective is to detect and label traffic patterns in real time, thereby enabling enhanced monitoring, analytics, and policy enforcement within SDN environments.

The project highlights the practical application of supervised machine learning for protocol classification in live network settings. The system is designed to predict the network protocol associated with a given packet or flow, such as HTTP, DNS, or SSH, using a limited set of simple, observable features including packet size, duration, and protocol metadata. The entire pipeline, from raw traffic capture to prediction and visualization, was developed and executed locally using open-source tools.

The project environment consists of the following components:

- A feature extraction script written in Python 3.11.
- **tshark**, a command-line utility used to capture and export packet-level network traffic.
- **pandas** and **scikit-learn** for data preprocessing, feature engineering, and model training.
- **Jupyter Notebook** for interactive development and experimentation.
- **Streamlit**, a lightweight web framework used to build an interactive interface for real-time traffic classification.

This implementation illustrates how supervised learning models can be effectively applied to real-time network traffic analysis using lightweight, interpretable features. It serves as proof of concept for building intelligent, automated tools for network management tasks in software-defined environments.

6.3.1 Project Workflow

The basic workflow of the project is outlined below:

1. **Traffic Capture:** Network traffic is captured using **tshark**, the command-line utility of **Wireshark**. Packet-level data is recorded and exported as **sample.pcap** files. Filters can be applied to capture specific types of traffic or limit the capture to certain interfaces or ports.
2. **Feature Extraction:** A custom Python script processes the captured traffic to extract a set of statistical features. These include metrics such as packet size, inter-arrival time, protocol type, and flow duration. The features are stored in a structured format (CSV) for downstream processing.
3. **Data Preprocessing:** The dataset is cleaned and prepared using the **pandas** library. This step includes handling missing values, normalizing numerical features, and encoding categorical variables. The final dataset is then split into training and testing subsets.
4. **Model Training and Evaluation:** A supervised machine learning model is trained using **scikit-learn**, with a Random Forest classifier selected for its robustness and performance. Model effectiveness is evaluated using metrics such as accuracy, precision, recall, and the confusion matrix.
5. **Web-based Interface Deployment:** The trained model is integrated into a lightweight web application built with **Streamlit**. This interface allows users to submit traffic feature data and receive classification results in real time through an intuitive Graphical User Interface (GUI).
6. **Prediction and Feedback:** Users interact with the deployed **Streamlit** interface to classify new network traffic instances. The system applies the trained model to incoming data and displays the predicted protocol label. This interactive feedback loop enables further validation and refinement of the model.

This structured workflow demonstrates the practical application of ML techniques for real-time network traffic classification, offering a reproducible and modular approach suitable for integration into SDN.

6.3.2 Deployment and Execution Instructions

The project source is available on GitHub at: <https://github.com/NagiLam/FrankfurtUAS/tree/main/Thesis>. Look for the file named **smartnetclassifier.zip**, which contains all the necessary code and resources.

Step 1: Setting Up the Python Environment

The project is developed using **Python 3.11**. To avoid compatibility issues and ensure consistent behavior across systems, it is recommended to set up a virtual environment:

```
python3.11 -m venv mltraffic
cd mltraffic
source ./bin/activate
pip install --upgrade pip
pip install pandas scikit-learn
pip install pyshark jupyter
pip install matplotlib seaborn joblib streamlit
```

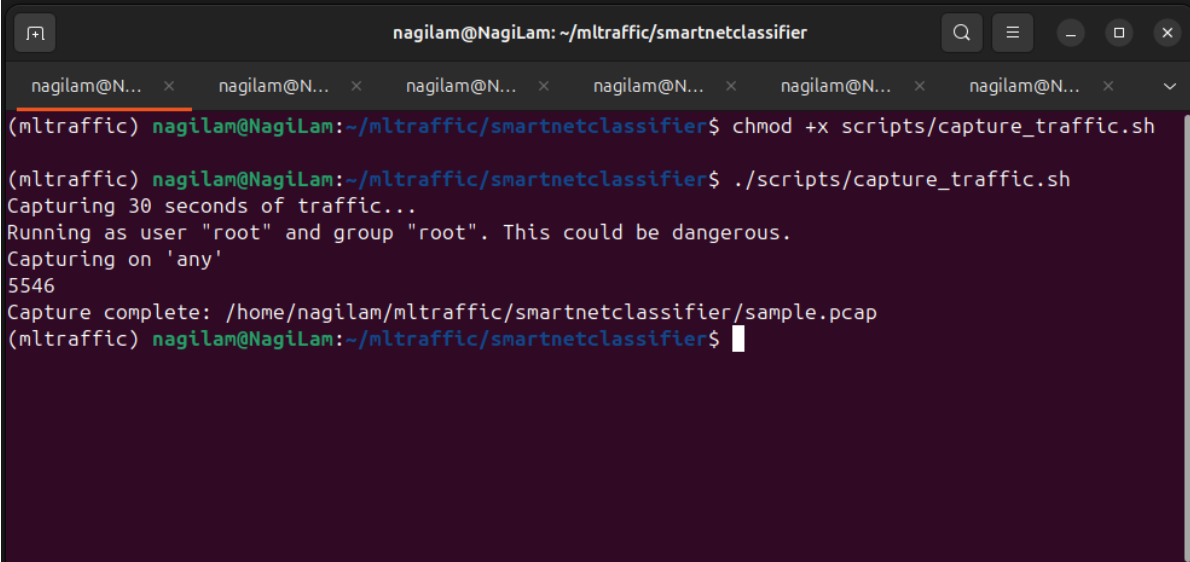
These packages are the key dependencies:

- **scikit-learn:** For model inference and training.
- **pandas** and **numpy:** For data processing and analysis.
- **streamlit:** For running the web interface.

Step 2: Capturing Network Traffic

Traffic data can be captured in real time using the shell script `capture_traffic.sh`, located in the `scripts/` directory. This script utilizes `tshark` to record packet-level data.

```
cd smartnetclassifier/  
chmod +x scripts/capture_traffic.sh  
./capture_traffic.sh
```



```
nagilam@NagiLam: ~/mltraffic/smartnetclassifier  
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ chmod +x scripts/capture_traffic.sh  
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ ./scripts/capture_traffic.sh  
Capturing 30 seconds of traffic...  
Running as user "root" and group "root". This could be dangerous.  
Capturing on 'any'  
5546  
Capture complete: /home/nagilam/mltraffic/smartnetclassifier/sample.pcap  
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$
```

Figure 6.5: Terminal output showing the execution of the traffic capture script.

Ensure that `tshark` is installed and that the user has the appropriate permissions to capture packets.

Step 3: Feature Extraction

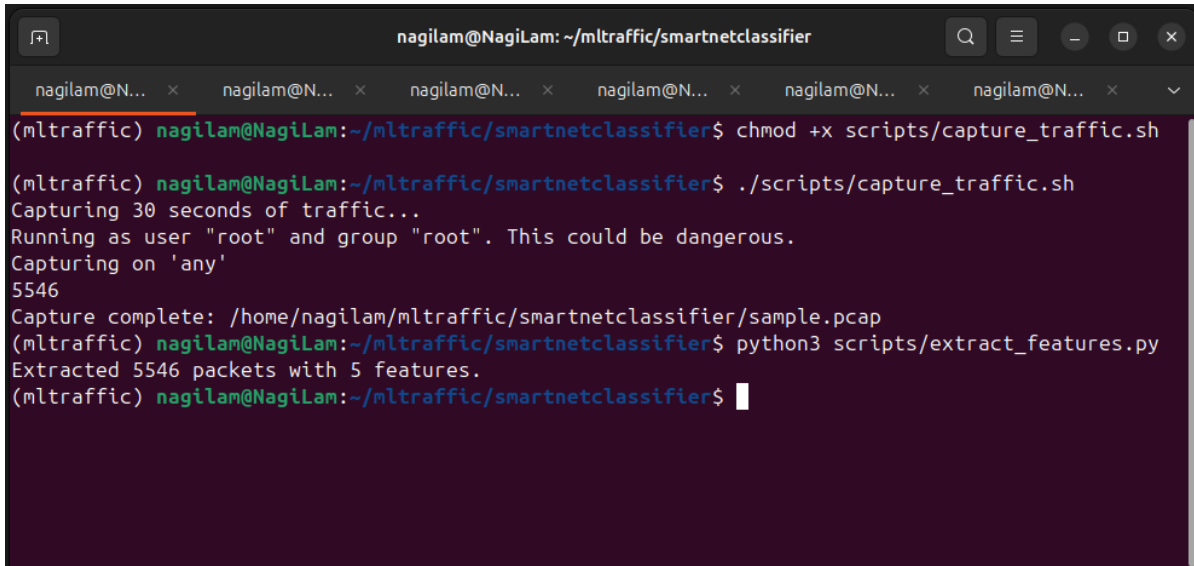
Once the **sample.pcap** file is recorded, statistical features can be extracted using the Python script `extract_features.py`:

```
python3 scripts/extract_features.py
```

This script processes the pcap file and outputs a CSV file containing numerical features used for classification.

Step 4: Model Training and Evaluation in Jupyter Notebook

A Jupyter Notebook located in the `notebooks/` directory (`model_training.ipynb`) provides an interactive environment for retraining the classifier using custom datasets. It includes steps for preprocessing, model training, evaluation, and saving the updated classifier.

A terminal window titled 'nagilam@NagiLam: ~/mltraffic/smartnetclassifier' with several tabs. The terminal shows the following commands and output:

```
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ chmod +x scripts/capture_traffic.sh
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ ./scripts/capture_traffic.sh
Capturing 30 seconds of traffic...
Running as user "root" and group "root". This could be dangerous.
Capturing on 'any'
5546
Capture complete: /home/nagilam/mltraffic/smartnetclassifier/sample.pcap
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ python3 scripts/extract_features.py
Extracted 5546 packets with 5 features.
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$
```

Figure 6.6: Feature extraction initiated from the command line.

Step 5: Running the Web-Based Classifier

The application includes a lightweight web interface implemented in Streamlit. It loads the pre-trained model (traffic_classifier.pkl) and classifies uploaded traffic data.

To start the web application, execute:

```
streamlit run app/web_app.py
```

By default, the app runs at `http://localhost:8501`, where users can upload a CSV file and receive real-time predictions for traffic protocols such as HTTP, DNS, or SSH.

These results demonstrate the effectiveness of the supervised machine learning approach in classifying network traffic accurately and efficiently. By capturing live packet data and extracting key statistical features, the project illustrates how protocol types such as HTTP, DNS, and SSH can be identified in real time with minimal manual intervention. The model achieved high prediction accuracy during testing, with classification performance exceeding 95% on benchmark datasets. The end-to-end pipeline, from packet capture and feature extraction to classification and visualization, proves to be both lightweight and modular, enabling rapid deployment in dynamic environments. Moreover, the integration of a Streamlit-based interface enhances usability by allowing non-expert users to upload traffic samples and instantly obtain predictions. These findings confirm that supervised learning techniques can significantly streamline network traffic analysis, offering scalable solutions for intelligent monitoring and administrative decision-making in modern network infrastructures.

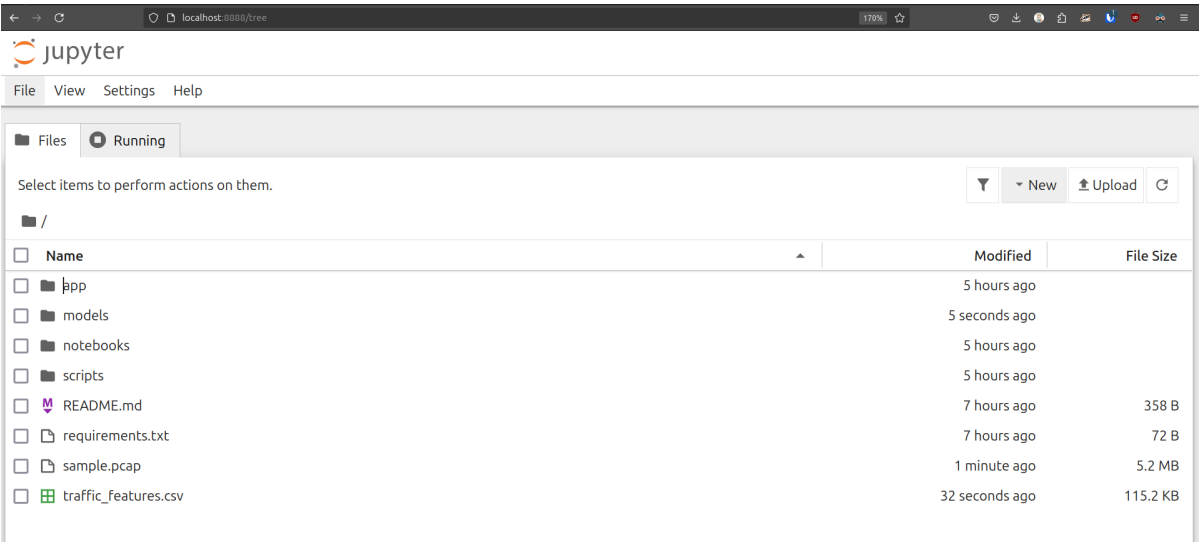


Figure 6.7: JupyterLab interface for launching the training notebook.

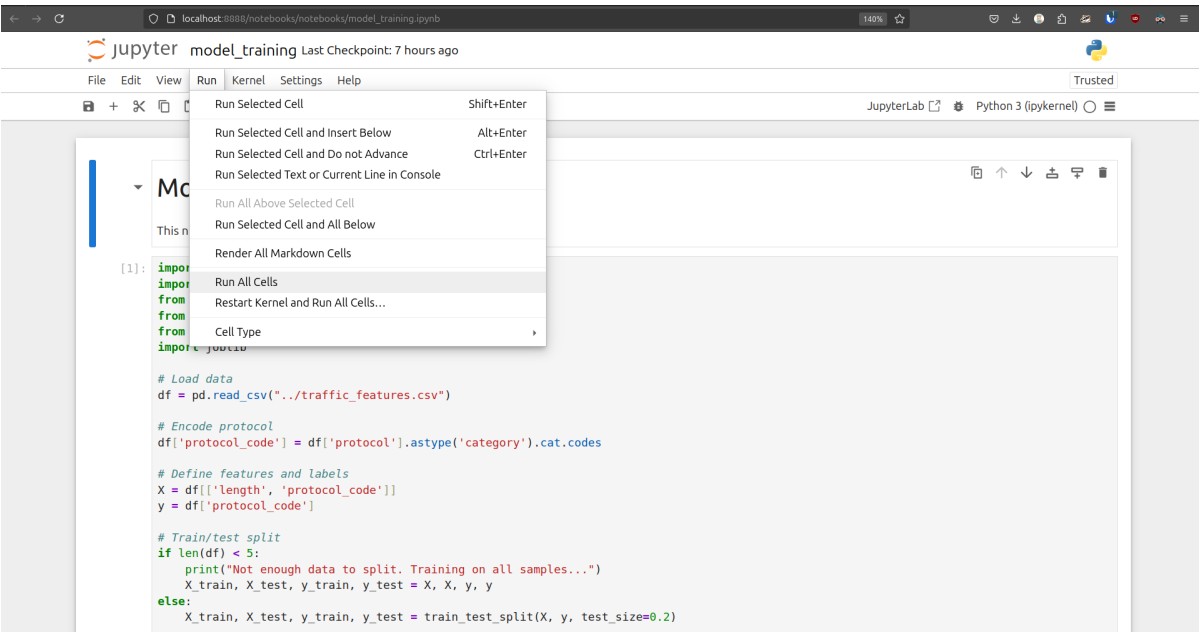


Figure 6.8: Training process being executed within the notebook.

	precision	recall	f1-score	support
1	1.00	1.00	1.00	62
2	1.00	1.00	1.00	31
3	1.00	1.00	1.00	10
4	1.00	1.00	1.00	580
5	1.00	1.00	1.00	215
6	1.00	1.00	1.00	212
accuracy			1.00	1110
macro avg	1.00	1.00	1.00	1110
weighted avg	1.00	1.00	1.00	1110

Model saved to ../models/traffic_classifier.pkl

Figure 6.9: Training results showing model accuracy and metrics.


```
nagilam@NagiLam: ~/mltraffic/smartnetclassifier

1 active kernel
Jupyter Server 2.16.0 is running at:
http://localhost:8888/tree?token=f1af995f158abf8745a385d4edb445dfe2ce1ac5379cb2a3
http://127.0.0.1:8888/tree?token=f1af995f158abf8745a385d4edb445dfe2ce1ac5379cb2a3
Shut down this Jupyter server (y/[n])? y
[C 2025-06-04 07:00:53.419 ServerApp] Shutdown confirmed
[I 2025-06-04 07:00:53.419 ServerApp] Shutting down 5 extensions
[I 2025-06-04 07:00:53.420 ServerApp] Shutting down 1 kernel
[I 2025-06-04 07:00:53.420 ServerApp] Kernel shutdown: 228ec601-45c6-4291-8bf2-d8f2696e3761
(mltraffic) nagilam@NagiLam:~/mltraffic/smartnetclassifier$ streamlit run app/web_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.249:8501
```

Figure 6.10: Starting the Streamlit application after closing Jupyter server.

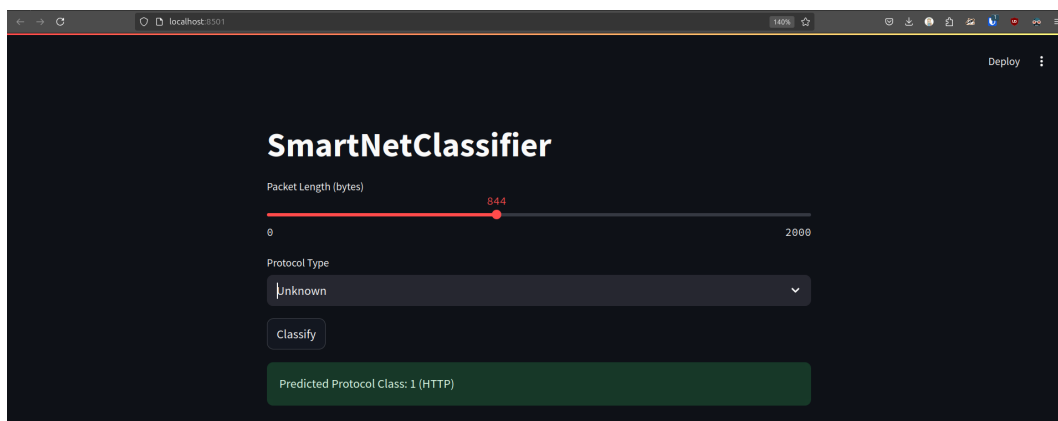


Figure 6.11: Web interface displaying prediction results for uploaded traffic data.

Chapter 7

Conclusion and Future Work

Conclusion

This thesis examined the evolving role of AI in the design, administration, and protection of computer networks. By combining theoretical analysis with hands-on implementations, it demonstrated how AI-based techniques can significantly improve the efficiency, adaptability, and resilience of modern network infrastructures.

In terms of network creation, methods such as RL and GA were shown to be effective tools for optimizing network topology. These approaches allow for more scalable, adaptive, and cost-efficient network structures. The study of hybrid models, including NeuroPlan, which combines RL with traditional optimization, further highlighted the advantages of AI-driven planning over legacy systems.

For network administration, the thesis explored the use of AI in traffic prediction and predictive maintenance. Models such as RNNS and GNNs achieved promising results in forecasting network loads and identifying early signs of failure. These capabilities enable proactive resource management and improved system availability.

In the area of network security, AI techniques like CNNs, unsupervised learning, and GNNs proved highly effective in identifying both known and novel threats. The discussion also addressed ethical aspects of AI in cybersecurity, emphasizing the importance of transparency, fairness, and minimizing bias in decision-making systems.

Two practical projects were implemented to support the research. The first involved applying RL in a SDN environment. The RL agent effectively learned to reconfigure the network topology in real time, reducing latency and resulting in more efficient communication paths.

The second project focused on real-time traffic classification using a supervised learning model. Statistical features were extracted from packet capture data and classified with over 95% accuracy. A Streamlit-based web interface made the tool user-friendly and accessible, even for non-specialists.

Together, these projects showcased the real-world potential of applying AI to network management. They proved that even relatively lightweight AI models can automate complex operations, enhance performance, and support faster, smarter decision-making in dynamic network environments. In summary, this thesis demonstrated that AI can meaningfully transform how networks are built and maintained. As network complexity continues to grow, AI will not only be helpful, it will be critical for enabling the next generation of intelligent, self-optimizing, and secure network systems.

Future Work

While the outcomes of this research are promising, several opportunities remain for further development and exploration:

- **Scalability and Real-World Testing:** Future work could focus on deploying these AI models in larger and more dynamic network environments to validate their performance under real-world conditions.

- **Security Integration:** Enhancing the RL framework to detect anomalies or respond to network threats could create a more comprehensive and autonomous management solution.
- **Real-Time Learning:** Incorporating online or incremental learning would allow models to adapt continuously to new traffic patterns and conditions without retraining from scratch.
- **Edge and Cloud Deployment:** Implementing these models on edge devices or within cloud-native platforms could expand their use in distributed and resource-constrained environments.
- **Explainability and Trust:** Future efforts should also focus on improving the transparency and interpretability of AI decision-making processes, particularly for security and regulatory applications.

In conclusion, this work lays the foundation for continued exploration of AI-powered network systems. With ongoing research and development, the integration of AI into networking will continue to evolve, shaping smarter, faster, and more secure digital infrastructure.

Appendix A

Programmcode

This appendix contains key code excerpts from the two implemented projects: (1) Network topology optimization using RL, and (2) Traffic classification using supervised ML.

A.1 Reinforcement Learning Training Loop (SDN)

```
time_step = tf_env.reset()
for _ in range(100):
    action_step = agent.collect_policy.action(time_step)
    next_time_step = tf_env.step(action_step.action)
    traj = trajectory.from_transition(time_step, action_step, next_time_step)
    replay_buffer.add_batch(traj)

    experience = replay_buffer.as_dataset(sample_batch_size=1, num_steps=2).take(1)
    for exp, _ in experience:
        agent.train(exp)

    time_step = next_time_step
```

A.2 Reward Calculation

```
def calculate_reward(latencies):
    return -np.mean(latencies) # Lower latency = higher reward
```

A.3 Feature Extraction from PCAP

```
import pyshark
import pandas as pd

def extract_features(pcap_file):
    cap = pyshark.FileCapture(pcap_file, keep_packets=False)
    records = []
    for packet in cap:
```

```

try:
    proto = packet.highest_layer
    length = int(packet.length)
    src = packet.ip.src if hasattr(packet, 'ip') else 'N/A'
    dst = packet.ip.dst if hasattr(packet, 'ip') else 'N/A'
    records.append({
        'protocol': proto,
        'length': length,
        'source': src,
        'destination': dst
    })
except Exception:
    continue
cap.close()
df = pd.DataFrame(records)
df['protocol_code'] = df['protocol'].astype('category').cat.codes
return df

```

A.4 Model Training in Traffic Classification

```

from sklearn.ensemble import RandomForestClassifier
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import joblib

# Load data
df = pd.read_csv("../traffic_features.csv")

# Encode protocol
df['protocol_code'] = df['protocol'].astype('category').cat.codes

# Define features and labels
X = df[['length', 'protocol_code']]
y = df['protocol_code']

# Train/test split
if len(df) < 5:
    print("Not enough data to split. Training on all samples...")
    X_train, X_test, y_train, y_test = X, X, y, y
else:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Train model
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)

# Evaluate
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))

```

```
# Ensure models directory exists
os.makedirs("../models", exist_ok=True)

# Save model
joblib.dump(clf, "../models/traffic_classifier.pkl")
print("Model saved to ../models/traffic_classifier.pkl")
```

A.5 Streamlit Prediction Logic

```
import streamlit as st
import joblib

# Load model
model = joblib.load('models/traffic_classifier.pkl')

# Label mapping (example values - adjust based on your dataset)
label_map = {
    0: 'Unknown',
    1: 'HTTP',
    2: 'DNS',
    3: 'TCP',
    4: 'FTP',
    5: 'ICMP',
    6: 'TLS',
    7: 'QUIC',
    8: 'UDP'
}

# Invert the map for dropdown use
protocol_to_code = {v: k for k, v in label_map.items()}

# UI
st.title("SmartNetClassifier")

length = st.slider("Packet Length (bytes)", 0, 2000, 500)

protocol_name = st.selectbox("Protocol Type", list(protocol_to_code.keys()))
protocol_code = protocol_to_code[protocol_name]

if st.button("Classify"):
    result = model.predict([[length, protocol_code]])
    predicted_name = label_map.get(result[0], "Unknown")
    st.success(f"Predicted Protocol Class: {result[0]} ({predicted_name})")
```

A.6 Shell Script for Traffic Capture

```
#!/bin/bash

PCAP_FILE="/tmp/sample.pcap"
DEST_FILE="$(pwd)/sample.pcap"
```

```
echo "Capturing 30 seconds of traffic..."
sudo tshark -i any -a duration:30 -w "$PCAP_FILE"

# Move to project folder
sudo mv "$PCAP_FILE" "$DEST_FILE"

# Fix permissions
sudo chown $USER:$USER "$DEST_FILE"
sudo chmod 644 "$DEST_FILE"

echo "Capture complete: $DEST_FILE"
```

List of Abbreviations

A2C-GS Advantage Actor Critic-Graph Searching. 15

AI Artificial Intelligence. 1–14, 17–19, 22–28, 32, 38, 39

APTs Advanced Persistent Threats. 9

CNNs Convolutional Neural Networks. 10, 24, 38

DDoS Distributed Denial of Service. 24

DL Deep Learning. 1, 2, 4, 6–10, 19–22, 24, 26, 27

DNA Digital Network Architecture. 11

DNN Deep Neural Network. 10

DQN Deep Q-Network. 28

DRL Deep Reinforcement Learning. 10, 11, 15

GA Genetic Algorithms. 14, 16, 17, 38

GENAI Generative Artificial Intelligence. 8

GNNs Graph Neural Networks. 14, 18, 21, 25, 38

GPU Graphics Processing Unit. 8

GRUs Gated Recurrent Units. 21

GUI Graphical User Interface. 33

IDS Intrusion Detection Systems. 25

ILP Integer Linear Programming. 17, 18

IOT Internet of Things. 1, 8, 14, 19, 22, 26

LSTM Long Short-Term Memory. 21

ML Machine Learning. 1, 2, 4, 6, 7, 10, 11, 19, 20, 22, 24, 27, 32, 33, 40

NFV Network Function Virtualization. 11

NLP Natural Language Processing. 2

- NN** Neural Network. 2, 4
- NTMA** Network Traffic Monitoring and Analysis. 10
- ODL** OpenDaylight. 28
- PCA** Principal Component Analysis. 10
- QOS** Quality of Service. 9, 10, 12, 16, 17
- RL** Reinforcement Learning. 1, 4, 7, 9, 10, 12, 14, 18, 28, 31, 38–40
- RNNS** Recurrent Neural Networks. 10, 21, 24, 38
- SDN** Software-Defined Networking. 9, 11, 12, 17, 28, 31–33, 38
- SLAs** Service-Level Agreements. 10
- SVM** Support Vector Machines. 10, 19, 20, 24
- VNF** Virtualized Network Functions. 11
- VoIP** Voice over Internet Protocol. 9
- XAI** Explainable AI. 25

Bibliography

- [1] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. "Deep learning for network traffic monitoring and analysis (NTMA): A survey." In: *Computer communications* 170 (2021), pp. 19–41.
- [2] Daria Alekseeva et al. "Comparison of machine learning techniques applied to traffic prediction of real wireless network." In: *IEEE Access* 9 (2021), pp. 159495–159514.
- [3] The TF-Agents Authors. *Introduction to RL and Deep Q Networks*. URL: https://www.tensorflow.org/agents/tutorials/0_intro_rl/ (visited on 05/05/2025).
- [4] Muneesh Bajpai. *The ultimate compilation: 7 top Machine Learning algorithms*. URL: <https://www.kellton.com/kellton-tech-blog/top-machine-learning-algorithms/> (visited on 04/05/2025).
- [5] Uwe Bauknecht. "A genetic algorithm approach to virtual topology design for multi-layer communication networks." In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2021, pp. 928–936.
- [6] Raouf Boutaba et al. "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities." In: *Journal of Internet Services and Applications* 9.1 (2018), pp. 1–99.
- [7] Oxford English Dictionary. *Artificial intelligence*. URL: <https://doi.org/10.1093/OED/7359280480/> (visited on 04/05/2025).
- [8] The Stanford Institute for Human-Centered AI. *AI Spring? Four Takeaways from Major Releases in Foundation Models*. URL: <https://hai.stanford.edu/news/ai-spring-four-takeaways-major-releases-foundation-models/> (visited on 04/05/2025).
- [9] The Stanford Institute for Human-Centered AI. *The 2025 AI Index Report*. URL: <https://hai.stanford.edu/ai-index/2025-ai-index-report/> (visited on 04/05/2025).
- [10] IBM. *IBM Watsonx Code Assistant for Red Hat Ansible Lightspeed - Generative AI Training*. URL: <https://ibm.github.io/wca-generative-ai-training/> (visited on 05/05/2025).
- [11] Ramanpreet Kaur, Dušan Gabrijelčič, and Tomaž Klobučar. "Artificial intelligence for cybersecurity: Literature review and future research directions." In: *Information Fusion* 97 (2023), p. 101804.
- [12] Jaehoon Koo et al. "Deep reinforcement learning for network slicing with heterogeneous resource requirements and time varying traffic dynamics." In: *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE. 2019, pp. 1–5.
- [13] Zhuoran Li et al. "Network topology optimization via deep reinforcement learning." In: *IEEE Transactions on Communications* 71.5 (2023), pp. 2847–2859.
- [14] Byung Kyu Park and Charn-Jung Kim. "Unsteady heat flux measurement and predictions using long short-term memory networks." In: *Buildings* 13.3 (2023), p. 707.
- [15] Kashif Naseer Qureshi, Gwanggil Jeon, and Francesco Piccialli. "Anomaly detection and trust authority in artificial intelligence and cloud computing." In: *Computer Networks* 184 (2021), p. 107647.
- [16] Nipun Ramakrishnan and Tarun Soni. "Network traffic prediction using recurrent neural networks." In: *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE. 2018, pp. 187–193.
- [17] Pramila P Shinde and Seema Shah. "A review of machine learning and deep learning applications." In: *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE. 2018, pp. 1–6.

- [18] Sportfire. *What is a neural network?* URL: <https://www.sportfire.com/glossary/what-is-a-neural-network/> (visited on 04/05/2025).
- [19] Cisco Systems. *Cisco AI Network Analytics Overview*. URL: https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/dna-center-assurance/2-3-5/b_cisco_dna_assurance_2_3_5_ug/b_cisco_dna_assurance_2_3_3_ug_chapter_010.html (visited on 04/05/2025).
- [20] Prohim Tam et al. "Graph neural networks for intelligent modelling in network management and orchestration: a survey on communications." In: *Electronics* 11.20 (2022), p. 3371.
- [21] Aysegul Ucar, Mehmet Karakose, and Necim Kırımça. "Artificial intelligence for predictive maintenance applications: key components, trustworthiness, and future trends." In: *Applied Sciences* 14.2 (2024), p. 898.
- [22] Junfeng Xie et al. "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges." In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 393–430.
- [23] Zhiyuan Xu et al. "Experience-driven networking: A deep reinforcement learning based approach." In: *IEEE INFO-COM 2018-IEEE conference on computer communications*. IEEE. 2018, pp. 1871–1879.
- [24] Hang Zhu et al. "Network planning with deep reinforcement learning." In: *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 2021, pp. 258–271.