People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Mohamed Khider University - Biskra

Faculty of Exact Sciences

Department of Computer Science

Thesis

Presented for the degree of **Doctor of Science**

Option : Network and Distributed Systems

Machine Learning Based Routing for the Internet of Things

Presented by: **Nour Elhouda Larouci**

Jury Committee:

President	Bennoui Hammadi	Professor	University of Biskra
Supervisor	Djeffal Abdelhamid	Professor	University of Biskra
Co-Supervisor	Sahraoui Somia	Associate Professor	University of Biskra
Examiner	Benharzallah Saber	Professor	University of Batna 2
Examiner	Ayad Soheyb	Associate Professor	University of Biskra

Publicly defended on: 02/11/2025

Dedication

To my parents,

who taught me patience, courage, and faith.

To my family,

whose unconditional support has been my source of strength.

To my friends,

for their encouragement, kindness, and companionship.

Nour El Houda Larouci

Acknowledgements

First and foremost, praises and thanks to **Allah**, the Most Gracious and the Most Merciful, for granting me the strength, patience, and perseverance to carry out and complete this research work.

I want to extend my sincere thanks and deep appreciation to my esteemed supervisor, **Professor Abdelhamid Djeffal**, for his insightful guidance, valuable expertise, and consistent availability throughout this academic journey. His vision, trust, and thoughtful feedback have been truly instrumental in shaping and enriching this work.

I also thank Dr. **Somia Sahraoui** for her collaboration and insightful discussions.

I am deeply grateful to my beloved **family**, especially my **parents**, siblings, and all those dear to me, for their unconditional love, constant encouragement, and heartfelt prayers. Their unwavering support has been the foundation of my strength, perseverance, and success.

I would also like to extend my sincere thanks to my **friends** and **colleagues** for their companionship, motivation, and kind words throughout this journey. Their presence has made this experience more meaningful and fulfilling.

Finally, I thank all those who contributed in any way to the realisation of this work.

Abstract

As the Internet of Things (IoT) continues to evolve and expand globally, it presents growing challenges for achieving efficient, scalable, and reliable data communication, particularly in mobile and resource-constrained environments. Mobile Ad hoc Network (MANET) routing protocols, commonly used in such settings, often rely on local information exchange and are not optimised for the dynamic and context-aware nature of smart city networks.

In this thesis, we are interested in enhancing routing efficiency in mobile IoT networks by incorporating learning-based decision models tailored to smart city environments, where mobility patterns can be leveraged to improve routing outcomes.

To this end, two model-based routing contributions have been proposed. The first introduces a Machine Learning-Based Routing Protocol (MLBRP) that uses historical routing information to train predictive models. Decision Trees (DT), Support Vector Machines (SVM), and Artificial Neural Networks (ANN) are employed to guide packet forwarding decisions. Simulation results show that MLBRP can reduce control message overhead by up to 73%, while enhancing energy efficiency, load balancing, and packet delivery performance.

The second contribution refines this approach by improving routing accuracy through deep learning. A Convolutional Neural Network (CNN)-based framework is proposed, enabling forwarding decisions based solely on local contextual and geographic information. By exploiting the regular mobility patterns observed in smart city scenarios, this model enhances path optimality, reduces end-to-end delay, and extends network lifetime.

Overall, the results demonstrate that machine learning, and particularly deep learning, offers strong potential for improving the intelligence, adaptability, and efficiency of routing in next-generation mobile IoT networks.

Key words: *Internet of Things, Smart City Networks, Machine Learning, Convolutional Neural Network, Routing Optimisation, Mobile Ad hoc Networks.*

Résumé

À mesure que l'Internet des Objets (IoT) continue d'évoluer et de se développer à l'échelle mondiale, il soulève des défis croissants en matière de communication de données efficace, évolutive et fiable, en particulier dans les environnements mobiles et limités en ressources. Les protocoles de routage utilisés dans les réseaux mobiles ad hoc (MANET), couramment déployés dans ces contextes, s'appuient généralement sur l'échange d'informations locales et ne sont pas optimisés pour la nature dynamique et contextuelle des réseaux urbains intelligents.

Dans ce travail de thèse, nous nous intéressons à l'amélioration de l'efficacité du routage dans les réseaux IoT mobiles, en intégrant des modèles décisionnels basés sur l'apprentissage automatique, adaptés aux environnements de villes intelligentes, où les schémas de mobilité peuvent être exploités pour optimiser les décisions de routage.

À cet effet, deux contributions basées sur des modèles ont été proposées. La première introduit un protocole de routage fondé sur l'apprentissage automatique (MLBRP), qui utilise l'historique des décisions de routage pour entraîner des modèles prédictifs. Des techniques telles que les arbres de décision (DT), les machines à vecteurs de support (SVM) et les réseaux de neurones artificiels (ANN) sont mobilisées pour guider la transmission des paquets. Les résultats de simulation démontrent que MLBRP permet de réduire le trafic de contrôle jusqu'à 73 %, tout en améliorant l'efficacité énergétique, l'équilibrage de charge et la performance de la livraison des paquets.

La seconde contribution perfectionne cette approche en renforçant la précision du routage grâce à l'apprentissage profond. Un cadre de routage basé sur un réseau de neurones convolutif (CNN) est proposé, permettant de prendre des décisions d'acheminement uniquement à partir des informations contextuelles et géographiques locales. En tirant parti des schémas de mobilité réguliers observés dans les scénarios de villes intelligentes, ce modèle améliore l'optimalité des chemins, réduit la latence de bout en bout et prolonge la durée de vie du réseau.

Dans l'ensemble, les résultats obtenus confirment que l'apprentissage automatique, et en particulier l'apprentissage profond, offre un fort potentiel pour renforcer l'intelligence, l'adaptabilité et l'efficacité des mécanismes de routage dans les réseaux IoT mobiles de nouvelle génération.

Mots-clés : *Internet des Objets, Réseaux de villes intelligentes, Apprentissage automatique, Réseau de neurones convolutifs, Optimisation du routage, Réseaux mobiles ad hoc.*

ملخص

مع استمرار تطور وانتشار إنترنت الأشياء (IoT) على المستوى العالمي، تبرز تحديات متزايدة لتحقيق اتصال بيانات فعّال، قابل للتوسعة وموثوق، لا سيما في البيئات المتنقلة والمقيدة بالموارد. غالبًا ما تعتمد بروتوكولات التوجيه في الشبكات المتنقلة اللاسلكية (MANET) المستخدمة في مثل هذه السياقات على تبادل المعلومات المحلية، لكنها لا تكون مُهيأة للتعامل مع الطبيعة الديناميكية والواعية للسياق في شبكات المدن الذكية.

تهدف هذه الأطروحة إلى تحسين كفاءة التوجيه في شبكات إنترنت الأشياء المتنقلة من خلال دمج نماذج اتخاذ القرار المعتمدة على التعلم الآلي، والمصممة خصيصًا لبيئات المدن الذكية حيث يمكن الاستفادة من أنماط الحركة المنتظمة لتحسين قرارات التوجيه.

ولهذا الغرض، تم اقتراح مساهمتين رئيسيتين مبنيتين على النماذج. المساهمة الأولى تقدم بروتوكول توجيه قائم على التعلم الآلي (MLBRP) يستخدم معلومات التوجيه التاريخية لتدريب نماذج تنبؤية. تم توظيف خوارزميات مثل أشجار القرار ،(DT) وآلات الدعم الناقل ،(SVM) والشبكات العصبية الاصطناعية (ANN) لتوجيه عملية تمرير الحزم. أظهرت نتائج المحاكاة أن البروتوكول المقترح يمكن أن يقلل من عبء رسائل التحكم بنسبة تصل إلى 73٪، مع تحسين كفاءة الطاقة، وتوازن الحمل، وكفاءة تسليم الحزم.

أما المساهمة الثانية، فتُحسن هذا النهج من خلال تعزيز دقة التوجيه باستخدام التعلم العميق. حيث تم اقتراح إطار عمل يعتمد على الشبكات العصبية الالتفافية ،(CNN) يسمح باتخاذ قرارات التوجيه اعتمادًا فقط على المعلومات السياقية والجغرافية المحلية. ومن خلال استغلال أنماط الحركة المنتظمة في سيناريوهات المدن الذكية، يُحسن هذا النموذج من مثالية المسار، ويقلل من زمن التأخير من طرف إلى طرف، ويُطيل عمر الشبكة.

بشكل عام، تؤكد النتائج أن تقنيات التعلم الآلي — وبشكل خاص التعلم العميق — تمتلك إمكانات قوية لتحسين ذكاء وكفاءة وقدرة التكيف في آليات التوجيه ضمن شبكات إنترنت الأشياء المتنقلة من الجيل القادم.

الكلمات المفتاحية: إنترنت الأشياء، شبكات المدن الذكية، التعلم الآلي، الشبكات العصبية الالتفافية، تحسين التوجيه، الشبكات المتنقلة اللاسلكية.

Contents

List of Figures

List of Tables

List of Abbreviations

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AODV	Ad hoc On-demand Distance Vector
CART	Classification and Regression Trees
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
DSR	Dynamic Source Routing
DT	Decision Trees
GANs	Generative Adversarial Networks
GNN	Graph Neural Networks
ID3	Iterative Dichotomiser 3
IPv6	Internet Protocol version 6
IoT	Internet of Things
LSTM	Long Short-Term Memory
MANET	Mobile Ad Hoc Network
MDP	Markov Decision Process
ML	Machine Learning
MLBRP	ML-Based Routing Protocol
MSE	Mean Squared Error
NAM	Network Animator
NFC	Near Field Communication
NS2	Network Simulator 2
OSPF	Open Shortest Path First

QoS	Quality of Service
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RFID	Radio Frequency Identification
RIP	Routing Information Protocol
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RREP	Route Reply
RREQ	Route Request
SDN	Software-Defined Networking
SVM	Support Vector Machine
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

List of Publications

International Journal Paper

- Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal, *"Machine Learning Based Routing Protocol (MLBRP) for Mobile Internet of Things Networks"*, *Journal of Network and Systems Management*, vol. 33, article no. 67, 2025. DOI: https://doi.org/10.1007/s10922-025-09949-6.

- Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal, *"Intelligent Packet Routing in IoT Networks Using a CNN-Based Deep Learning Approach"*, *Wireless Networks*, (Submitted).

International conference paper

- Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal, *"A Survey of Deep Learning Solutions for Network Routing in Modern Networks: Challenges and Opportunities"*, in *Proceedings of the 2025 International Symposium on iNnovative Informatics of Biskra (ISNIB)*, IEEE, pp. 1–6, 2025. DOI: https://doi.org/10.1109/ISNIB64820.2025.10983118.

Chapter book

- Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal, *"Machine Learning for Routing in IoT: A Review"*, in *Recent Advances in Communication Technology, Computing and Engineering*, edited by Mariyam Ouaissa, Mariya Ouaissa, Sarah El Himer, and Zakaria Boulouard, RGN Publications, pp. 195–207, 2021. ISBN: 978-81-954166-0-8, DOI: https://doi.org/10.26713/978-81-954166-0-8.

General introduction

1 Context

The Internet of Things (IoT) [1] is a transformative technological paradigm that envisions the pervasive interconnection of physical objects, from household appliances and vehicles to critical industrial machinery and urban infrastructure, through the Internet. This vision extends the scope of the digital world beyond traditional computing devices to include "smart" objects capable of sensing, processing, communicating, and autonomously reacting to their environment. By embedding computation and connectivity into the physical world, the IoT paradigm enables real-time data collection, intelligent analysis, and automated decision-making, thus offering promising applications across various domains such as smart cities, intelligent transportation systems, healthcare, agriculture, and environmental monitoring.

The impact of IoT is particularly visible in urban settings, where it is a cornerstone of smart city initiatives [2]. Here, interconnected devices coordinate to enhance transportation efficiency, optimise energy consumption, improve public safety, and deliver citizen-centric services. Vehicular Ad hoc Networks (VANETs), for instance, enable real-time communication between vehicles and infrastructure, facilitating applications such as traffic flow optimisation, autonomous vehicle coordination, and accident prevention. Similarly, drone-enabled IoT networks are revolutionising fields like precision agriculture and large-scale environmental data collection [3].

Despite the increasing adoption and diverse applications of IoT, the underlying communication infrastructure faces significant challenges. The exponential increase in the number of connected devices has led to a massive surge in data traffic [4]. This situation exposes the limitations of traditional network protocols, especially in dynamic and mobile scenarios. IoT environments are inherently characterised by resource constraints [5]: devices typically have limited processing power, memory, and energy capacity, and are often deployed in settings where the network topology changes rapidly and unpredictably.

One of the most essential functions in any network, including IoT, is packet routing. It refers to the process of selecting the most appropriate path for data to travel from a source node to its destination. In networks where node positions and connections remain fixed over time, routing is relatively straightforward and can often be handled by conventional algorithms. In contrast, IoT networks introduce a set of unique challenges. Devices in these environments are often resource-constrained in terms of processing power, memory, and energy. Moreover, they are frequently deployed in scenarios where the network topology changes dynamically due to node mobility, intermittent connectivity, or environmental factors, which significantly affects routing performance [6]. These conditions make routing in IoT not only more complex but also highly dynamic and context-dependent.

In response to these challenges, research has explored various routing protocols, especially those adapted from Mobile Ad hoc Networks (MANETs) [7]. While these protocols, such as the Ad hoc On-Demand Distance Vector (AODV), have served as foundational techniques, they were not designed with IoT's scale, mobility patterns, or energy constraints in mind . As the number and heterogeneity of connected devices grow, these protocols often struggle to meet the demands of IoT, such as low power availability, dynamic mobility patterns, and scalability requirements [8, 9].

In smart city environments, the mobility of humans and devices often follows predictable routines [10, 11, 12]. Commuters, for example, exhibit temporal regularity in their daily routes. These recurring patterns offer valuable, yet underutilized, information that can be leveraged to improve routing decisions and overall network performance.

In light of these challenges and emerging opportunities, Machine Learning (ML) [13] and Deep Learning (DL) techniques offer a promising path forward. These data-driven methods can uncover complex relationships and temporal patterns from historical network data, enabling models to generalise and make informed decisions under changing conditions. When applied to networking, ML can enhance the routing process by leveraging knowledge from past routing outcomes to make adaptive, efficient decisions in real time, particularly in highly dynamic environments.

Advances in edge computing [14, 15], model optimisation [16], and data availability have made it increasingly practical to deploy ML and DL models on network nodes, even in resource-constrained IoT environments. These models can be trained to recognise traffic trends [17], anticipate node movements [18, 19], and select optimal forwarding paths with reduced reliance on frequent control message exchanges. This shift toward intelligent, data-driven routing represents a significant evolution from traditional protocols, offering greater

adaptability and performance in real-world conditions.

This thesis is motivated by the need to bridge the gap between traditional routing methods and the adaptive, predictive capabilities offered by modern ML techniques, particularly in the context of mobile IoT networks in smart cities.

2 Problem statement

Despite the promising potential of IoT, its practical deployment is hindered by several fundamental networking challenges, particularly in the area of packet routing. These challenges can be summarised as follows:

- **Inefficiency of traditional routing protocols:** Protocols like AODV and DSR, initially developed for MANETs, are not well suited for large-scale, energy-constrained, and highly dynamic IoT environments. They typically depend on extensive broadcasting of control messages for route discovery and maintenance. This process is repeated each time a route is needed, even under similar conditions, resulting in excessive signalling overhead and poor resource utilisation.

- **Absence of intelligent decision-making:** Classical routing approaches rely on predefined rules and do not learn from past routing outcomes. They overlook valuable information such as node behaviour, environmental dynamics, or traffic characteristics, which could be used to improve routing efficiency.

- **Neglect of mobility regularity:** In many smart city scenarios, the mobility of nodes follows regular and structured patterns. However, conventional protocols fail to exploit this predictable movement, missing an opportunity to anticipate route changes and reduce unnecessary route discoveries.

- **High energy consumption and limited scalability:** Traditional routing approaches often rely on repeated broadcasting of control messages for route discovery and continuous route maintenance. This behaviour consumes substantial energy, which is a critical issue for battery-powered IoT devices, and it also negatively affects the scalability of the network, especially in dense and highly mobile settings.

- **Lack of adaptive learning mechanisms**: Most existing enhancements to routing protocols still follow predefined rules and lack the ability to dynamically adapt based

on observed network behaviour or historical routing success. This limitation restricts their ability to respond intelligently to changing network conditions.

Overcoming these limitations requires a shift toward intelligent, data-driven routing solutions that can adapt based on observed network behaviour and regular mobility patterns. By integrating learning mechanisms into the routing process, IoT networks can become more efficient, scalable, and responsive to real-world challenges.

3 Contributions

To address the previously discussed problems, this thesis presents two main contributions aimed at enhancing the intelligence and adaptability of the routing process in IoT networks using ML and DL techniques.

Contribution 1: Machine learning based routing protocol (MLBRP) for mobile IoT networks

The first contribution proposes an ML-Based Routing Protocol (MLBRP) designed to support intelligent routing decisions in mobile IoT environments. The key features of this contribution include:

- A lightweight, distributed ML model deployed at each node, trained on past routing decisions and context-related observations to support informed next-hop selection.
- Prediction of the optimal next hop without relying on broadcasting of control messages.
- Exploitation of regular mobility patterns and recurring routing conditions, as commonly observed in smart city scenarios.
- Reduction of control overhead, contributing to better energy efficiency and improved scalability in dynamic networks.

The MLBRP leverages classical ML techniques such as Decision Trees (DTs), Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs) to construct predictive models, trained on simulated routing scenarios.

Contribution 2: Intelligent packet routing in IoT networks using a CNN-based deep learning approach

Building on the foundation laid by MLBRP, the second contribution presents a DL-based enhancement using a Convolutional Neural Network (CNN) for intelligent packet routing. This approach leverages DL's ability to automatically learn complex patterns from high-dimensional data, improving routing performance in dynamic IoT scenarios. Key advancements in this contribution include:

- Construction of an extensive dataset, both in sample volume and feature diversity, to meet DL training requirements and improve model generalisation.
- Design of a CNN architecture comprising convolutional and pooling layers for automated feature extraction, followed by fully connected layers for accurate next-hop prediction.
- Elimination of repeated control message exchanges by enabling routing decisions based solely on locally observed contextual and spatial information.
- Exploitation of the regularity of mobility patterns, a key property of smart city environments, within the model's learning process to guide optimal next-hop selection.
- Improved model accuracy and routing reliability, demonstrating superior performance compared to classical ML models, particularly in complex smart city network scenarios.

4 Dissertation plan

This thesis is organised into five chapters as follows:

Chapter 1: Presents an overview of IoT, including its enabling technologies, layered architecture, and application domains. It highlights the importance of routing in IoT networks, classifies common routing protocols, and discusses key performance metrics and the main challenges encountered in dynamic, resource-constrained environments.

Chapter 2: Provides a detailed overview of ML and DL techniques relevant to intelligent routing. It covers traditional ML algorithms such as DT, SVM, and ANN, as well as CNNs designed to enhance decision-making based on spatial and temporal patterns.

Chapter 3: Reviews existing works related to routing in mobile and dynamic networks, identifying the limitations of conventional protocols and highlighting the growing use of ML and DL techniques to address challenges such as mobility and energy efficiency. It covers a range of intelligent routing approaches and outlines research gaps that motivate this thesis.

Chapter 4: Details the first contribution of this thesis: a routing protocol based on classical ML models (DT, SVM, and ANN). It explains how local context and prior routing experiences are leveraged for next-hop prediction and describes the training methodology, simulation setup, and performance evaluation.

Chapter 5: Describes the second contribution: a CNN-based DL routing strategy aimed at enhancing next-hop prediction accuracy. It covers the design of input features, the training process, and the evaluation of the proposed model within a smart city mobility scenario.

Chapter 1

Internet of Things

1.1 Introduction

IoT is a modern technological paradigm that connects everyday physical objects, embedded with sensors, software, and communication capabilities,to the internet, enabling them to collect and exchange data autonomously. This interconnected environment enables real-time monitoring, intelligent decision-making, and automation across various domains, including healthcare, transportation, smart homes, and industrial systems.

IoT is characterised by the integration of intelligent, self-configuring, and resource constrained nodes into a global network infrastructure. These nodes operate in dynamic and often constrained environments, raising critical challenges related to reliability, scalability, energy efficiency, security, and privacy. Despite these constraints, IoT enables pervasive computing and continues to revolutionise the way devices interact and respond to their surroundings.

This chapter provides an overview of the essential components of IoT systems, including enabling technologies, device-level functionalities, architectural layers, and practical applications. It also addresses the importance of routing in IoT networks, offering a classification of routing protocols, performance evaluation metrics, and the main challenges faced in ensuring efficient data transmission in resource-limited environments.

1.2 Definition

IoT is a new pervasive-computing paradigm allowing a wide variety of "things" to be directly connected to the Internet [20]. The term "things" refers to a broad and flexible set of entities,

including physical devices, watches, home appliances, vehicles, animals, and even people, all of which can connect and provide services through the Internet infrastructure (see Figure 1.1). These things are equipped with connectivity modules, software, sensors, and actuators to adapt their connection to the Internet world. Things gain intelligence when additional features, such as sensors and processing power, are added. This allows them to interact with their environment, gather information, and make appropriate decisions without human intervention. This new evolution of the Internet aims to offer end users and the environment smarter services that will enhance their quality of life in every way.

Simply put, the IoT is a network that connects everyday objects and devices to the Internet, allowing them to send and receive data and be monitored, controlled, and automated from a distance.

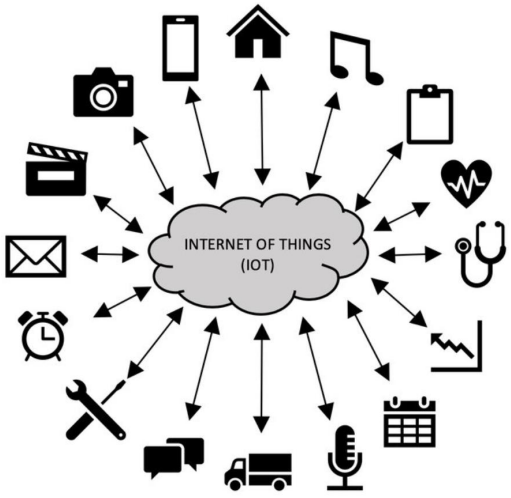

Figure 1.1: IoT representative diagram [21]

1.3 Building blocks and enabler technologies

To achieve the idea behind the IoT paradigm, which is enabling all sorts of physical objects to share information and coordinate decisions and also transform these objects from ordinary objects to smart things, many technologies and elements are woven together intricately to result in this new Internet evolution. In the upcoming section, we will explore seven fundamental elements essential for ensuring the proper functioning of the IoT, as illustrated in Figure 1.2. The key elements [22] of the IoT include:

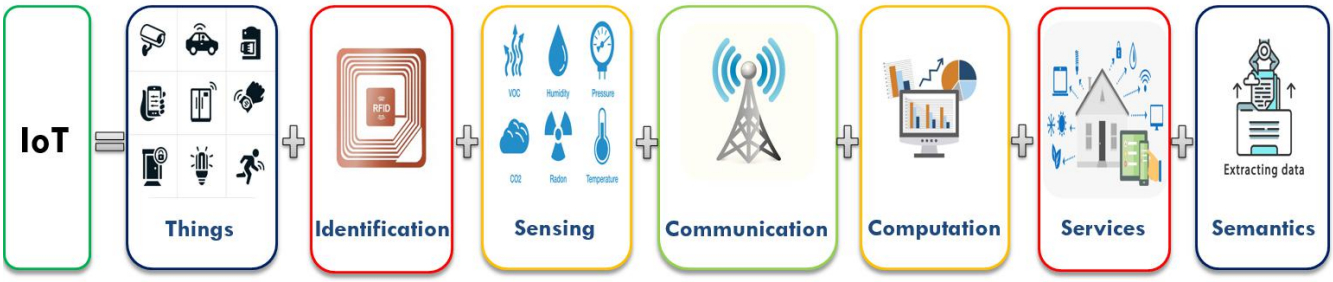

Figure 1.2: The IoT elements

1.3.1 IoT device (The Thing concept)

The IoT represents a revolutionary technological advancement, introducing a paradigm shift in connecting various objects to the Internet and facilitating intelligent interactions and collaborations between these devices. This innovative idea goes beyond connectivity, enabling seamless communication and collaboration between various devices. The term "connected things" refers to a broad scope of entities. At one end of the spectrum, we find simple sensors and actuators that can capture and transmit data, providing valuable insights into various aspects of our environment. These sensors can monitor temperature, humidity, light, motion, and numerous other parameters, crucial in environmental monitoring systems, building automation, and smart city initiatives. Moving further along the spectrum, we encounter various complex devices with integrated IoT capabilities. Smart appliances have become increasingly prevalent, enabling remote control, automation, and energy optimisation in homes and businesses. These appliances include intelligent refrigerators, thermostats, lighting systems, and security cameras, among others, enhancing convenience, energy efficiency, and security in our daily lives. Additionally, wearable devices have emerged as a significant category within the IoT ecosystem. These devices, such as smartwatches, fitness trackers, and health monitoring devices, integrate with our lives, tracking our physical activity, heart rate, sleep patterns, and more. By providing personalised data and actionable insights, wearable devices empower individuals to make informed decisions about their well-being and lifestyle. The industrial sector has also witnessed a transformative impact through IoT adoption. Industrial machinery equipped with IoT capabilities allows for remote monitoring, predictive maintenance, and optimisation of operational processes. This integration enhances efficiency, minimises downtime, and enables proactive decision-making within manufacturing plants, supply chains, and logistics networks. Furthermore, IoT has extended its reach to encompass a broader spectrum of entities, including living beings. By incorporating IoT technologies

into human and animal tracking systems, health monitoring, and safety applications, we can ensure the well-being of individuals, livestock, and wildlife. IoT-enabled devices like biometric sensors and GPS trackers enable real-time monitoring and timely emergency response.

1.3.2 Identification

Identification [23] plays a crucial role in the functioning of an IoT network as it provides a unique identity for every object present within the network. This identity allows the network to differentiate between various objects and facilitates effective communication and interaction among them. Identification encompasses two main processes: naming and addressing. Naming involves assigning a distinctive name to each object in the network. This name serves as a human-readable label or identifier for the object, making it easier for users or administrators to refer to and identify specific objects. On the other hand, addressing involves assigning a unique address to each object in the network. This address serves as a machine-readable identifier that allows devices and systems within the network to locate and communicate with a particular object. Addressing is essential for establishing reliable and efficient communication pathways within the IoT network. In the realm of IoT, there are several techniques available for identifying devices, such as Radio Frequency Identification (RFID) and ubiquitous ID (uID) [24]. These methods provide distinct codes or identifiers that can be associated with IoT devices, enabling their identification within the network. Additionally, addressing methods for IoT objects often involve using Internet Protocol version 6 (IPv6). IPv6 [25] is a widely adopted network protocol that provides a large address space and various addressing features suitable for IoT deployments. However, concerning low-power wireless networks, such as those found in IoT applications, the overhead of IPv6 headers can be a concern. This is where IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) comes into play. 6LoWPAN [26] is a protocol that offers a compression mechanism for IPv6 headers. By compressing the headers, 6LoWPAN reduces their size, making IPv6 addressing more suitable for low-power wireless networks. This compression mechanism eliminates redundant information and utilises compact representations for IPv6 header fields, optimising the utilisation of limited network resources in low-power environments.

1.3.3 Sensing and monitoring

By incorporating Wireless Sensor Network (WSN) techniques, IoT technology will improve the ability to monitor the environment efficiently. WSN [27] consists of a collection of tiny,

low-cost, low-power, multifunctional sensor nodes. They are responsible for covering a particular geographical area, gathering the sensed information autonomously, and sending it to a warehouse-of-collection called the sink node. The sensor nodes interact with each other to sense sensitive data, such as temperature, humidity, pressure, light, motion, gas, sound, and location. This collaboration between the sensors is called the sensing process. Regarding IoT technology, IoT devices are equipped with sensors to act intelligently, interact with the environment, and perceive and monitor their surroundings. Thus, IoT sensing plays a crucial role in the larger ecosystem of IoT applications by capturing real-time or periodic data from various objects and leveraging that information for decision-making, optimisation, automation, and enhanced services. It enables the creation of intelligent systems that can respond dynamically to environmental changes, improving efficiency, productivity, and convenience in various domains.

1.3.4 Communication

IoT devices play a crucial role in gathering data by sensing and monitoring the environment. Once these devices capture the necessary information, they must transmit it to a central sink node for further processing and analysis. To achieve this, communication becomes vital as IoT devices collaborate with each other to exchange data effectively. The primary method of communication between IoT devices is connecting them to the global network, commonly known as the Internet Infrastructure. This connectivity allows IoT devices to leverage the vast resources and reach offered by the internet. IoT devices can establish connections and share information by utilising various communication technologies. Several technologies are available to facilitate communication in IoT systems. RFID enables the identification and tracking of objects using radio waves, while Near Field Communication (NFC) allows for short-range wireless communication between devices. Bluetooth technology provides wireless data transfer over short distances, Wi-Fi enables high-speed wireless communication within local networks, and Long Term Evolution (LTE) supports long-range wireless communication with high data transfer rates. These technologies empower IoT devices to communicate and collaborate effectively, forming a network connecting diverse objects and delivering specific intelligent services. In the context of IoT communication, devices need to operate with low power consumption, considering the constraints imposed by IoT deployments. IoT nodes often operate on limited power resources and must maintain efficiency even in the presence of lossy and noisy communication links. This requirement ensures that IoT devices can function reliably and sustainably in various environments.

1.3.5 Computation

Computation refers to the ability of IoT devices to process data and perform calculations. This is accomplished through processing units such as microcontrollers, microprocessors, System-on-Chip (SOC), Field-Programmable Gate Arrays (FPGAs), and software applications. These components serve as the "brain" of the IoT system, allowing it to perform computational tasks. To support IoT applications, several hardware platforms have been developed, including Arduino, UDOO, FriendlyARM, Intel Galileo, Raspberry Pi, Gadgeteer, BeagleBone, Cubieboard, Z1, WiSense, Mulle, and T-Mote Sky. These platforms provide the infrastructure required for IoT devices to perform computational functions and data processing tasks. Another critical computational component in the IoT ecosystem is cloud computing. This platform provides facilities for intelligent objects to send data to the cloud for real-time big-data processing. Cloud computing provides the computational power and storage capabilities required to handle the massive amounts of data generated by IoT devices.

1.3.6 Services

IoT services are divided into four main categories: Identity-related Services, Information Aggregation Services, Collaborative-Aware Services, and Ubiquitous Services. Identity-related services are fundamental and required for other services because they involve identifying real-world objects in the virtual world. Information Aggregation Services gather and summarise raw sensory measurements for processing and reporting to IoT applications. Collaborative-aware services extend Information Aggregation Services by using the data collected to make decisions and respond appropriately. Ubiquitous Services seeks to provide collaborative ware services whenever and wherever they are required. The ultimate goal of IoT applications is to provide universal services, but significant obstacles exist.

1.3.7 Semantics

The term "semantic" in the IoT context refers to machines' ability to extract knowledge intelligently to provide the required services. Knowledge extraction entails locating and utilising resources, as well as modelling data. It also includes data recognition and analysis to make informed decisions and provide precise services. Semantic Web technologies such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL) support this requirement. Furthermore, Efficient XML Interchange (EXI) is designed to optimise XML applications for resource-constrained environments. EXI reduces bandwidth and minimises

storage requirements by converting XML messages into a binary format. Additionally, artificial intelligence (AI) techniques, such as ML and DL, enable IoT devices to learn from data, make intelligent decisions, and even exhibit autonomous behaviour. These technologies empower IoT systems to adapt, optimise, and improve performance over time.

1.4 Architecture

IoT technology connects a wide and varied range of things through the Internet. This explains why layered architecture is needed. The IoT architecture serves as a structured framework that outlines how different components of the IoT ecosystem interact with each other. It provides a blueprint or guidelines that ensure coherent integration and interoperability between various IoT devices, networks, and platforms. IoT architecture does not have a single consensus that is accepted by all. Various researchers have proposed layered architectures: three, four, and five layers [23]. However, according to many researchers [22], the IoT mainly operates on three layers, termed the perception, network, and application layers as depicted in Figure 1.3. Hereinafter, an explanation of each layer, including the devices and technologies encompassed within each one.

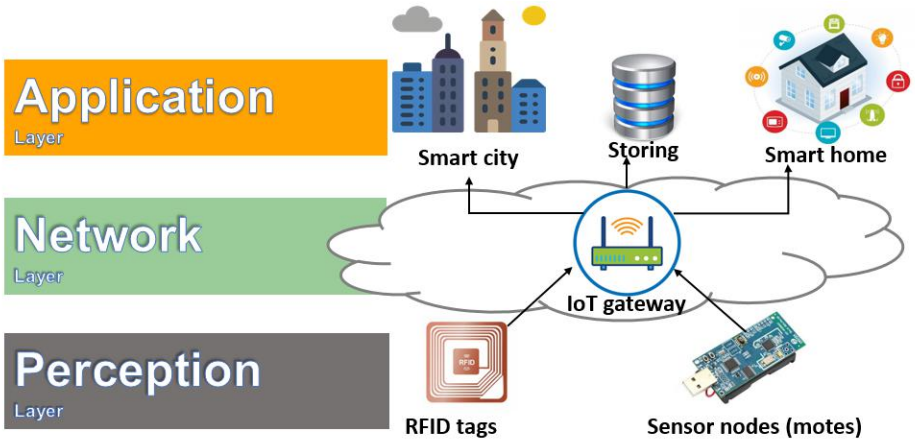

Figure 1.3: IoT basic architecture [28]

1.4.1 Perception layer

The perception layer [29], often called the sensing or data acquisition layer, can be considered the physical foundation of the entire IoT ecosystem. Its primary role revolves around serving as the sensory apparatus of IoT devices, playing an essential role in environment

monitoring and data collection. This layer constitutes the first point of contact between the physical world and the digital realm, bringing tangible reality into the digital domain. At the heart of the perception layer lies a myriad of sensors, each designed to perform specific tasks. Depending on the intended application, these sensors come in various forms, such as temperature, humidity, light, motion detectors, and many others. Their purpose is to gather environmental information, sense critical physical parameters, and even identify and communicate with other smart objects within their vicinity. After the perception layer has diligently gathered data, it must be processed and refined before it becomes useful information. This data preprocessing involves tasks like filtering out noise, aggregating readings, and, in some cases, converting analogue data to digital format. Following preprocessing, the refined data is then transmitted to the next layer, typically the network or communication layer.

1.4.2 Network layer

This intermediate layer [23], also known as the connectivity layer, serves as a bridge between the foundational perception layer, responsible for data acquisition through sensors, and the application layer. Its primary role is to ensure all sensing things' connectivity and allow them to communicate and share data with other connected smart things. It accomplishes this by employing a spectrum of connectivity technologies, each tailored to suit specific use cases and requirements. These technologies include:

- Wi-Fi: it is a prevalent and versatile connectivity option, suitable for IoT devices that require high-speed data transfer, such as smart home appliances and security cameras. It provides a stable and relatively long-range wireless connection within a local area.
- Bluetooth: this technology is excellent for short-range, low-power connections. It is commonly used in wearable devices, smart home peripherals, and proximity-based applications.
- Zigbee: it is well-suited for low-power, low-data-rate applications that require robust and scalable mesh networking. It's frequently used in home automation, industrial automation, and smart lighting systems.
- Cellular Networks: cellular connectivity options encompass 2G, 3G, 4G, and ultra-fast 5G networks. These are ideal for IoT devices that need to transmit data over long distances or remain connected while in motion, such as fleet management systems, asset tracking, and smart city infrastructure.

The versatility of these connectivity technologies allows IoT ecosystems to be adaptable and responsive to diverse scenarios and environments. Moreover, the network layer must be equipped with gateways or access points, which act as intermediaries between IoT devices and the broader network infrastructure, such as the Internet. These gateways often perform essential functions like protocol translation, data aggregation, and security enforcement. Beyond ensuring connectivity, the network layer takes on a vital role in data management. It handles the routing and transfer of data between the edge devices and the cloud or other centralised platforms where data analysis and decision-making occur. Efficient data routing is essential, especially when dealing with a vast number of IoT devices concurrently sending data.

1.4.3 Application layer

The application layer [23] represents the uppermost tier of the IoT architecture, and its primary mission is to receive and process the data transmitted from the lower layers, particularly the network layer. The received data are used to provide: (i) appropriate services and operations tailored to meet the needs of the end-users and customers. It addresses diverse deployment scenarios and applications for the IoT, including smart homes, smart cities, and intelligent agriculture; (ii) data services, such as data storage (backup services), data processing, data analysis, and data mining. In addition to these functions, the application layer is responsible for user interfaces and interaction, presenting data and insights to end-users through web applications, mobile apps, or dashboards. It also manages user authentication, access control, and authorisation to ensure data privacy and security.

1.5 Applications and use cases

As IoT technology develops, it opens the door for cutting-edge applications that will revolutionise industries, increase productivity, improve efficiency, and enhance our daily lives. These applications [30] encompass a variety of fields, including smart homes, healthcare, transportation, industry, agriculture, manufacturing, and more. In the following section, we will elaborate on the primary application domains that are commonly utilised by people daily. Additionally, we will provide various scenarios to illustrate their use cases (see Figure 1.4).

1.5.1 Smart homes

IoT devices can control and automate various aspects of a home, such as lighting, temperature, security systems, and appliances, to improve its occupants' convenience, security, energy efficiency, and general comfort. Users can monitor and control these devices remotely using smartphones, tablets, or voice commands [31]. The following are some of the key views of smart home technologies and their benefits:

- Home automation
  Smart homes use automation to control lighting and temperature. Homeowners can schedule lights to turn on or off at particular times and adjust thermostats remotely to reduce energy use.

- Energy efficiency
  Smart homes attempt to decrease energy use. Smart thermostats, for example, are IoT devices that can learn the patterns of occupants and adjust heating and cooling as necessary to reduce energy consumption. When no one is in the room, smart lighting systems can automatically turn off the lights, and smart appliances can run at off-peak times to save money on energy.

- Security and safety
  Smart home security systems provide remote-monitored surveillance cameras, video doorbells, and smart locks. Home security is improved because residents can view live feeds and receive alerts. In an emergency, smart Carbon monoxide (CO) and smoke detectors can send notifications and trigger alarms.

1.5.2 Healthcare

Healthcare [29] aims to enhance patient care and enable remote management and monitoring of medical systems and equipment. IoT devices, such as sensors, wearables, and smart medical devices, are integrated into the healthcare ecosystem to collect, transmit, and analyze real-time data for various healthcare purposes. IoT technologies offer the healthcare industry many advantages. Thus, instead of visiting the doctors periodically, the IoT uses devices worn by patients and specialised sensors injected into their bodies to track them and collect information in real-time relating to medical parameters (heartbeat, blood pressure, blood sugar, etc.). The collected data are transferred to a distant server, allowing doctors to monitor patients in serious situations remotely. Here are some key use cases:

- Patient monitoring and tracking
  IoT wearables like smartwatches and fitness trackers can continuously monitor vital signs such as heart rate, blood pressure, and activity levels. These devices can alert healthcare providers and patients to any anomalies or changes in health, enabling early intervention.

- Identification and authentication
  IoT can incorporate biometric sensors like fingerprint scanners or facial recognition for secure patient identification. This helps in preventing medical identity theft and ensures that patient data remains confidential. Moreover, IoT-enabled medication dispensers leverage these authentication mechanisms to verify a patient's identity before administering the prescribed medication dosage. This proactive approach significantly reduces the potential for medication errors, enhancing patient safety and the overall quality of healthcare delivery.

- Remote consultations
  IoT-enabled Video Conferencing: IoT can facilitate remote consultations with healthcare providers through secure video conferencing solutions. Patients can receive medical advice without the need for in-person visits, making healthcare more accessible, especially in remote areas.

1.5.3 Transportation and logistics

The transportation and logistics domain [30] is increasingly integrating IoT applications to enhance sustainability, efficiency, and safety. Modern vehicles, such as cars, buses, trains, bicycles, and railways, are now equipped with sensors, actuators, and advanced computing capabilities. Moreover, tags and sensors deployed on roads and transported goods optimize traffic routing, aid in depot management, provide essential travel information to tourists, and monitor the status of transported goods. This data is transmitted to traffic control centers and transportation vehicles, facilitating seamless operations.

These advancements illustrate how IoT technologies are transforming transportation:

- Connected vehicles
  The IoT makes it easier for vehicles to communicate with one another and the surrounding infrastructure.

- Vehicle-to-Vehicle (V2V) communication
  Vehicles share data such as speed and location to avoid accidents and enhance the efficiency of traffic movement.

- Vehicle-to-Infrastructure (V2I) communication
  Vehicles establish communication with infrastructure alongside the road, such as traffic lights or parking systems, to enhance traffic management and minimize congestion.

- Intelligent traffic control systems
  A centralized system connects and synchronises traffic lights and signals, allowing them to adjust timings in response to real-time traffic conditions dynamically. This intelligent adjustment helps alleviate congestion on the roads.

- Public transport management
  IoT devices play a role in effectively managing public transportation systems by offering up-to-date details on timetables, vehicle positions, and passenger movement.

- Parking sensors
  IoT sensors identify the presence of available parking spaces and relay this information to drivers, directing them to the closest unoccupied spot.

- Smart parking
  IoT applications enhance parking availability and minimise the time spent searching for vacant parking spaces.

- Real-time vehicle tracking
  By integrating IoT devices into vehicles, it becomes possible to track their location, speed, and route in real time. This capability enables improved fleet management and optimisation.

- Maintenance and diagnostics
  Engine performance and health data are gathered through vehicle sensors, enabling predictive maintenance and reducing downtime.

- Logistics
  real-time information processing technology, employing RFID and NFC, allows for constant monitoring of the entire supply chain. This includes design, procurement, production, transportation, storage, distribution, sales, returns, and after-sales service. By

providing timely and accurate product information, this technology enables enterprises and supply chains to respond to dynamic market conditions efficiently.

1.5.4 Smart cities

IoT plays a vital role in developing smart cities by integrating diverse elements in urban infrastructure, such as public services, utilities, buildings, transportation systems, and citizens. This connectivity allows for the real-time gathering of data, its analysis, and informed decision-making, resulting in more efficient resource management and an elevated standard of living for the residents. Furthermore, Smart cities can be considered a valuable solution for reducing urban waste and pollution. According to the UNU-EGOV, Smart cities are defined as a continuous and evolving process that depends on utilising information and communication technology (ICT) to facilitate change. This process involves the active participation of all stakeholders and encourages citizens to actively contribute towards enhancing the quality of life and promoting sustainability [32]. The adoption of IoT technology in modern cities can be viewed in different examples. The following are examples of IoT applications in smart cities.

- Water management can be enhanced through the utilisation of IoT sensors, which can oversee the quality of water, identify leaks, and optimise the distribution of water networks. This technological advancement aids in the preservation of water resources and the mitigation of water wastage.

- Environment surveillance
  IoT sensors can measure air quality, noise levels, and weather conditions to provide valuable data for environmental monitoring. This information can be used to implement appropriate measures and policies to improve the overall environmental quality.

- Waste management
  Smart waste management systems use IoT sensors to monitor bin garbage levels and optimise waste collection routes. This approach minimises unnecessary pickups, reduces costs, and keeps the city cleaner.

- Public safety
  IoT-enabled surveillance cameras, connected emergency response systems, and smart street lighting enhance public safety. Real-time data and video analytics can help detect and respond to incidents quickly.

- Energy management
  IoT devices can monitor and control energy consumption in buildings, streetlights, and other infrastructure. This enables efficient energy usage, reduces costs, and promotes sustainability.

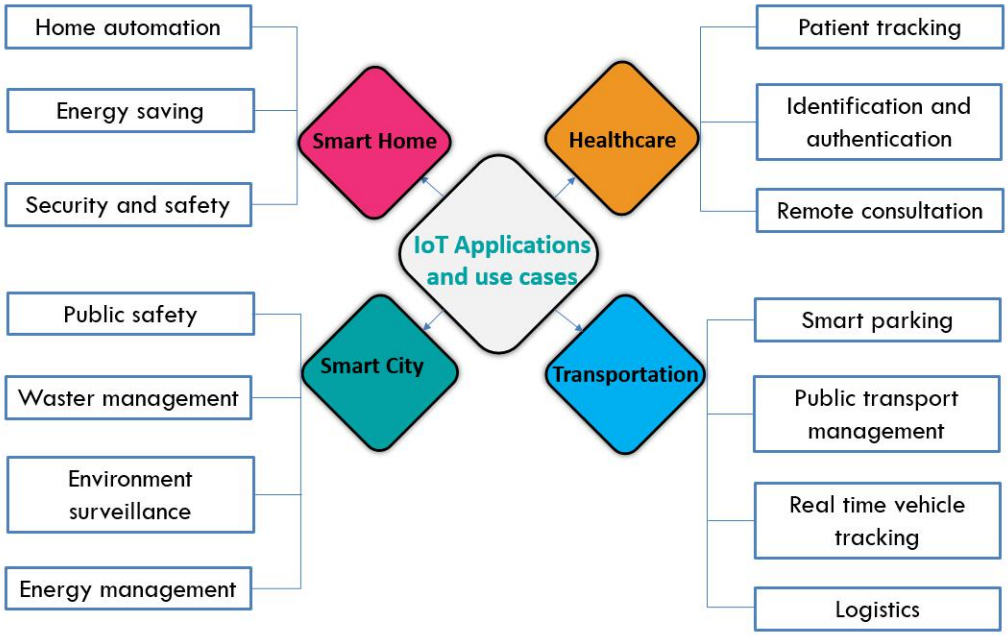

Figure 1.4: IoT-relevant application domains and use cases [33]

1.6 Routing in the Internet of Things

1.6.1 Overview of the routing problem

Routing generally addresses the question of how to determine a network path or even multiple paths that data packets traverse from its origin to its intended endpoint. Considering that the source and the destination are not always directly connected physically, this imposes that data packets be relayed from one intermediate node to another before reaching the destination. This strategy is commonly referred to as multi-hop routing (see Figure 1.5). The sequence of hops or the intermediate nodes participating in forwarding the data packet is referred to as a routing path or route [34]. Routing algorithms play a critical role in selecting a specific path for data transmission. These algorithms take into consideration various factors to determine the optimal route for data packets. Different routing protocols have been developed, as explained in section 1.6.3, to accommodate the diverse nature of

existing networks. The choice of one routing protocol over another depends on several factors, including the network's topology, the level of reliability required, and the specific goals of the network (e.g., minimising latency, optimising for bandwidth, or enhancing fault tolerance).

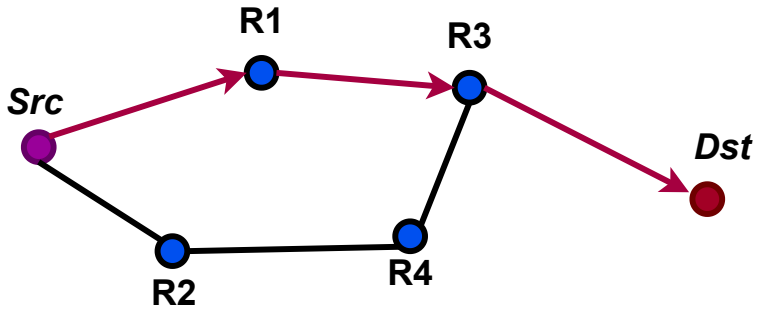

Figure 1.5: The general routing problem [34]

1.6.2 Routing in IoT

At an IoT scale, the routing of information generated by Things takes place in two major domains: IoT networks and the Internet, as depicted in Figure 1.6. Routing inside the Internet is mainly based on the IP stack, which is mature, well-documented, and has been working well in practice. In IoT networks, the routing is based on the 6LoWPAN standard, specifically designed for low-power IoT devices, allowing them to communicate using IPv6 [34]. Some IoT networks use a "border router" as an intermediary device to bridge between the IoT network and the internet. A border router manages the communication between IoT devices and the external network. However, it's important to note that IoT networks can employ various communication methods, including Wi-Fi, cellular, LPWAN (Low Power Wide Area Network), and others. In some cases, IoT devices themselves can connect directly to the internet without the need for a border router.

IoT devices establish connectivity with the internet through a border router, leveraging IPv6 addressing and the 6LoWPAN protocol. IPv6 is commonly used in IoT networks due to its larger address space and suitability for connecting a vast number of devices. IoT devices may be assigned IPv6 addresses, which can be either public or private. Some IoT devices may be assigned public IPv6 addresses, making them directly reachable from the broader internet. Public addresses are typically used when devices need to be globally accessible, such as IoT devices deployed in smart cities or public infrastructure. Whereas many IoT deployments use

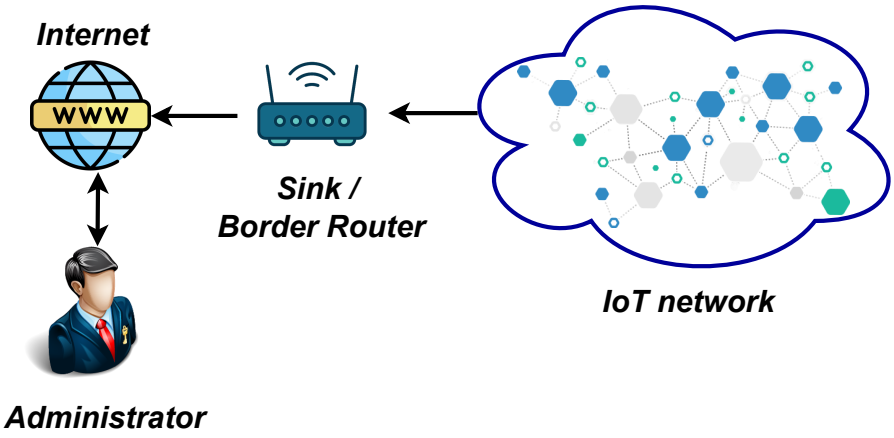

Figure 1.6: An illustrative example of an IoT system [35]

private IPv6 addresses within their local networks. These addresses are not directly routable on the public internet but are used for communication within the local IoT network. The border router plays a significant role in bridging the gap between private IPv6 addresses and the public Internet. On the other side, 6LoWPAN is a protocol designed to enable the use of IPv6 over low-power wireless networks, which are common in IoT deployments. It allows IoT devices with limited resources to use IPv6 addressing and communication efficiently. IoT devices use the 6LoWPAN protocol to adapt IPv6 packets to the constraints of low-power wireless networks. This includes compressing IPv6 headers to reduce overhead and fragmentation to fit data into smaller packets.

Furthermore, an IoT network consists of several hundred or even thousands of devices. The latter are deployed on a large scale to monitor the environment and measure its physical parameters. Typically, IoT devices are resource-constrained in terms of processing power, power supply, memory, and the lossy link network. Therefore, the routing task raises many challenges inherent to their particular characteristics. First, the limited resources impose an optimised use of them to extend the network lifetime. Second, the sensors are disseminated in an ad-hoc way and must self-organise to form the network and route the information. Third, a huge amount of data is generated by these constrained nodes, which requires being routed in an optimised manner. Fourth, the design of a sensor network protocol can be closely related to the target application. For example, the need to convey monitoring information from an industrial site or a nuclear power plant requires a limited latency, which is different from that tolerated by a conventional temperature reading application (building monitoring) [36].

1.6.3 Routing protocols classification

Routing protocols are essential for determining the optimal paths data packets should take from a source to a destination. These protocols are classified into various categories based on their operational characteristics and network architecture (see Figure 1.7). In the context of the IoT, routing protocols play a vital role in enabling efficient communication among a multitude of interconnected devices. IoT routing protocols are typically classified into several categories based on their suitability for the unique characteristics and requirements of IoT networks. The classification of routing protocols helps network administrators choose the most suitable routing method for their specific network environment. Subsequently, we will explain the different categories of IoT routing protocols and provide insights into their characteristics.

1.6.3.1 Based on route selection [4]

A. Proactive protocols

Proactive routing protocols establish and maintain routes to all possible destinations within the network continuously (before any packets are sent). This protocol maintains a routing table, which is periodically updated based on a fresh destination list. This involves continuous and periodic data transmission. Hence, it is known as a table-driven protocol. Examples of proactive routing protocols include Open Shortest Path First (OSPF) and Routing Information Protocol (RIP). Proactive routing protocols are generally less suitable for many IoT applications due to their constant route maintenance. Maintaining routing tables for all possible destinations can be impractical in large-scale IoT deployments, where devices are numerous and resource-constrained. However, proactive protocols can be used in smaller-scale or more stable IoT networks. For instance, in smart homes or building automation systems, where device locations and connections are relatively stable, proactive routing can work efficiently. Routing tables are maintained with low overhead in such cases.

B. Reactive protocols

Reactive routing protocols, also known as on-demand routing protocols, create paths only in response to an actual data transmission request from a source to a destination. These protocols do not maintain a constant, complete routing table but instead, establish routes dynamically in response to specific data transfer requirements. Reactive routing is more bandwidth-efficient when compared to proactive routing, but it can introduce latency as routes are determined on demand. Popular examples of reactive routing protocols include

AODV and DSR. These protocols are frequently a better fit for IoT networks. IoT devices frequently run on batteries and have limited resource availability. It is more effective in these environments to create routes only when data transmission is required. Reactive routing reduces the overhead associated with maintaining routing tables, which is crucial for energy conservation in battery-powered IoT devices. Reactive routing is advantageous in IoT applications such as environmental monitoring, agriculture, or wildlife tracking where data transmission is intermittent.

C. Hybrid protocols

Hybrid routing protocols provide a balance between proactive and reactive approaches. They are flexible and can adapt to various IoT scenarios. In IoT, hybrid routing protocols are used in situations where some devices require continuous communication (proactive) while others need on-demand routing (reactive). For example, in a smart city, where streetlights are continuously monitoring and sending data while emergency response sensors only activate during specific events, a hybrid approach could efficiently cater to the diverse communication needs of these devices.

1.6.3.2 Based on network structure [34]

A. Flat-based routing

The flat-based routing, also known as flat routing, employs an equally functioning fashion for all network nodes, assigning them identical sensing tasks. However, this uniformity may lead to data redundancy as multiple nodes collect and transmit similar information, resulting in substantial energy consumption. Consequently, the overall lifetime of battery-powered IoT devices may suffer, posing challenges in energy-sensitive applications. Thus, this kind of routing protocol is good for small networks and offers high reliability.

B. Hierarchical routing

Due to the high density of the sensors, a flat routing can lead to the overloading of the base station or that of nearby nodes (hotspot phenomenon). This overload would cause increased latency and excessive energy consumption. To remedy this and cope with the heavy traffic load without degrading the QoS, the network is subdivided into clusters [36]. Unlike flat routing, nodes have different functions and roles. The main idea is that of reserving the share of energy consumption devoted to multi-hop communications within the cluster. This greatly reduces the number of communications in the direction of the sink. Within each cluster,

a designated node, known as the cluster head, takes responsibility for aggregating cluster information before routing it to the base station. Depending on factors like performance, network scale, or the chosen routing protocol, the cluster head may communicate with the base station in a single hop or via multi-hop routes that involve other cluster heads. The constitution of clusters and the selection of cluster heads contribute to improving scalability, the energy consumption of nodes, and the overall lifetime of the IoT network.

C. Location-based routing

In a location-based routing protocol, information about the location of the routing nodes is used to address nodes and forward data packets. The nodes' locations can be obtained via dedicated hardware (e.g., GPS sensors) or software (e.g., location discovery algorithms). The forwarding decision is usually based on a distance metric (e.g., Euclidean distance). The distance between neighbouring nodes can be estimated based on incoming signal strengths. Relative coordinates of neighbouring nodes can be obtained by exchanging such information between neighbours. Alternatively, the location of nodes may be available directly by communicating with a satellite using GPS if nodes are equipped with a small, low-power GPS receiver. To save energy, some location-based schemes demand that nodes go to sleep if there is no activity. More energy savings can be obtained by having as many sleeping nodes in the network as possible.

1.6.3.3 Based on protocol operation [37]

A. Multi-path based routing

Multipath routing is a fault-tolerant routing technique that involves establishing multiple communication paths between the source and destination nodes in a network. Instead of relying on a single path for data transmission, multipath routing utilises several alternative paths concurrently. If a failure occurs along one path, the data can be rerouted through an alternative path, ensuring continuous data transmission. Multipath routing enhances the network's robustness and reliability by reducing the impact of single-point failures and improving load balancing.

B. Query-based routing

Query-based routing in the context of the IoT refers to a data retrieval approach where a central node or application, acting as the sink, sends queries to specific IoT devices or sensor nodes in the network to gather relevant information. These queries typically include specific

data parameters and conditions to filter the desired data. When the IoT devices receive the query, they evaluate whether they meet the criteria and respond accordingly, providing the requested data or notifying the sink about the data's availability and location. This query-based mechanism optimises data collection by selectively retrieving essential information, reducing unnecessary data transmission, and conserving network resources. It enables real-time and efficient data gathering in IoT applications, facilitating timely decision-making and enhancing overall system performance. Moreover, query-based routing's adaptability allows the sink node to dynamically adjust its data collection needs, accommodating the changing requirements of IoT applications and providing a scalable and flexible solution for diverse IoT deployments.

C. Negotiation-based routing

Negotiation-based routing is a communication strategy that aims to enhance resource management and efficiency in IoT networks. As IoT devices are often resource-constrained and operate in dynamic environments, efficient utilisation of network resources becomes crucial. The protocols that belong to this category propose transmitting a series of negotiation messages before transmitting real data, to prevent redundant information from circulating in the network. Negotiation decisions are made based on the availability of resources.

D. QoS-based routing

In Quality of Service (QoS)-based routing protocols, the underlying network faces the intricate challenge of achieving a delicate equilibrium between conserving energy resources and ensuring the delivery of data with high-quality performance. A crucial aspect of this balancing act involves adhering to specific QoS metrics, such as minimising data transmission delay, optimising energy consumption, and effectively managing available bandwidth. The ultimate objective is to efficiently and reliably route data to the designated Base Station (BS) while simultaneously meeting the stringent requirements set forth by these QoS metrics. Consequently, the network must intelligently allocate its resources and dynamically adapt its routing decisions to guarantee that data delivery to the BS is not only timely and energy-efficient but also meets the predefined QoS criteria. As such, QoS-based routing protocols play a crucial role in maximising the overall efficiency and performance of the network, ensuring that data transmission to the BS is consistently accomplished with the maximum effectiveness and adherence to the specified QoS metrics and contributing to the success of various IoT applications and services.

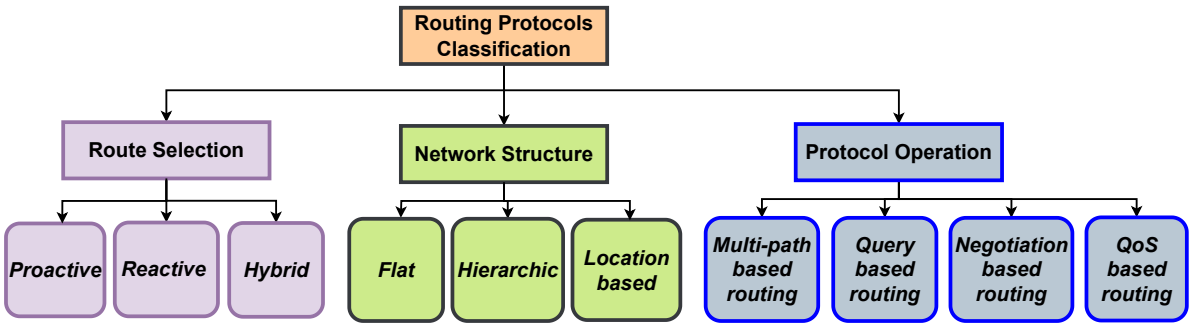

Figure 1.7: Taxonomy of IoT routing protocols [38]

1.6.4 Routing metrics that influence the choice of the routing protocol

The performance of an IoT network is closely tied to the selection of an appropriate routing protocol and the associated routing metrics. Routing metrics are numerical parameters used by routing algorithms to determine the most suitable path for data packets to travel from source to destination within the network. These metrics are typically categorised into two main types: node metrics and link metrics.

- **Node metrics** pertain to the characteristics of individual network nodes. These characteristics may include factors such as energy capacity, antenna sensitivity, and expected node lifetime.
- **Link metrics**, on the other hand, provide information about the quality of the connections (links) between nodes. Common link metrics include measurements of throughput, delay, error bit rate, link quality, and other channel-specific characteristics.

The choice of routing protocol and the specific metrics used should align with the requirements and constraints of IoT applications to optimise network performance.

1.6.5 Performance measures

The Internet serves as the basic infrastructure required by distributed applications running on various end systems. In an ideal scenario, we would hope for Internet services to transfer any amount of data between any two end systems instantly and without any data loss. Unfortunately, this is a lofty objective that cannot be realised in the real world. In reality, computer networks inherently limit the amount of data that can be transferred between end

systems per second (throughput), introduce delays in communication, and may lead to the loss of data packets [39]. In IoT context, it's crucial to recognize that the Internet serves as the fundamental framework upon which services are built for various distributed applications running on end systems. The ultimate goal is to have Internet services that can effortlessly transmit large amounts of data between any two end systems instantly without experiencing any data loss. Nevertheless, achieving this ambitious goal in practice is challenging to achieve in practicality. In reality, computer networks, including the Internet, impose specific constraints on the transmission of data between these end systems, and these limitations become evident in various ways. In the following section, we will delve into the essential metrics, namely throughput, end-to-end delay, and packet loss and remaining energy.

A. Throughput

Throughput refers to the rate at which data can be successfully transmitted from one point to another within a network. In the context of IoT, it represents the amount of data that can be transferred per unit of time, typically measured in bits per second (bps) or a similar unit. Throughput is influenced by various factors such as network bandwidth, congestion, and the efficiency of routing protocols. In IoT applications, achieving sufficient throughput is crucial to support the timely exchange of data between devices, sensors, and central systems.

B. End-to-end delay

End-to-end delay, also known as latency, is the time it takes for data to travel from the source to the destination across a network. It includes various components, such as processing delays, queuing delays, transmission delays, and propagation delays. In the IoT context, minimising end-to-end delay is important for applications that require real-time interactions, like industrial automation or remote control systems. High latency can lead to delays in data delivery, which might be unacceptable for time-sensitive tasks.

C. Packet loss

Packet loss occurs when data packets being transmitted through a network fail to reach their intended destination. This can happen due to network congestion, interference, signal degradation, or errors in the transmission process. In IoT networks, where data integrity is crucial for accurate decision-making and control, packet loss can lead to incomplete or inaccurate information. Effective error detection and correction mechanisms are employed to minimise the impact of packet loss in IoT communications.

D. Remaining energy

Remaining energy refers to the amount of power or battery capacity left in an IoT device. Many IoT devices are battery-powered and operate in resource-constrained environments. Monitoring the remaining energy is vital to ensure uninterrupted operation and prevent unexpected device failures. In routing decisions within IoT networks, considering the energy levels of nodes becomes essential to optimise communication paths and extend the network's overall lifetime.

1.6.6 Routing challenges

Routing in IoT networks faces several unique challenges due to the characteristics of IoT devices and the nature of IoT environment. Routing protocols must be able to efficiently transmit data while meeting certain performance criteria. The major challenges that directly affect routing in the IoT are presented below.

A. Limited resources

Reducing the physical size of IoT devices (such as a sensor node or an RFID tag) has resulted in limited resources encompassing energy, processing power, memory, and communication capabilities. In a typical sensor node, the microcontroller is equipped with limited processing power (typically a few MIPS), a small program memory (a few kilobytes), and a relatively modest general-purpose memory (usually a few hundred kilobytes). These resource limitations necessitate a high level of optimisation and simplification on IoT devices. Energy conservation takes on paramount importance for IoT devices because they are often mass-produced, deployed in large quantities, and expected to operate autonomously for extended periods. This makes replacing the energy supply, such as the battery, for each IoT device an exceptionally challenging task. Consequently, it becomes essential to optimise processing tasks. Furthermore, in IoT ecosystem, all devices must be connected to the network infrastructure and be capable of wireless communication. Given that wireless communication typically consumes a significant portion of an IoT device's energy, it is imperative to minimise its usage to the greatest extent possible [34].

B. Node deployment

An essential component of IoT network design is node deployment, especially as the number of nodes grows. An efficient node deployment [36] approach can significantly reduce several scale-related difficulties, including communication, routing efficiency, and energy consump-

tion. When nodes are strategically placed, communication distances are minimised, which results in improved signal quality and reduced transmission power requirements. Routing becomes more efficient as a well-planned deployment results in a structured network topology, reducing the overhead associated with maintaining routes. Additionally, balanced energy consumption, achieved through optimal node positioning, extends the operational life of battery-powered devices. The consideration of node deployment as a primary concern in routing protocol design is essential for ensuring the scalability, reliability, and energy efficiency of IoT networks, making it a foundational element in the overall success of the network architecture, particularly when dealing with larger node populations.

C. Node mobility

Node mobility [36] in IoT networks introduces a dynamic and constantly evolving network topology, leading to the challenge of maintaining stable and efficient communication paths as devices move. Routing data to and from these moving nodes becomes complex due to the uncertainty caused by mobility. This dynamic nature also necessitates frequent updates, resulting in increased data traffic, which in turn can consume more energy, especially in resource-constrained devices. The interplay of mobility, routing challenges, and the energy demands of frequent updates underscores the need for specialized protocols that can adapt to changing topologies, manage the efficient routing of data to moving nodes, and strike a balance between the benefits of dynamic information exchange and the energy constraints of the network, thus ensuring reliable and sustainable operation in the face of node mobility.

D. Fault tolerance

Fault tolerance [36] is crucial in IoT systems due to the inherent vulnerability of devices to various failure scenarios, including power supply depletion, communication link errors, interference, and hardware damage. These failures can disrupt the normal operation of the network and compromise its overall performance. When one or more nodes fail, it's essential that the network can adapt to these unexpected events to minimise the impact. Robust routing protocols play a vital role in this context by incorporating mechanisms to detect failures, dynamically reroute data along alternative paths, and prevent data loss. These protocols ensure that the network remains resilient, data continues to flow, and devices can maintain communication even in the face of node failures. This resilience ensures that the IoT system can continue to function reliably and deliver the expected services, making fault tolerance a critical consideration in IoT network design.

E. Quality of service

In IoT, the QoS [36] requirement is paramount for specific applications that demand precise performance levels to maintain the value and integrity of the transmitted data. QoS encompasses critical parameters such as low delay (minimal transmission latency), high throughput (adequate data transfer capacity), low jitter (consistent and predictable packet arrival times), and a high packet delivery ratio (minimal data loss). These parameters directly impact the usability of the information delivered. For instance, in real-time applications like remote patient monitoring or industrial automation, ensuring low delay is essential to receive timely updates, while low jitter is crucial for maintaining synchronisation in data streams. If QoS is not upheld, data may arrive too late, become inconsistent, or even be lost, rendering the information irrelevant or unreliable, which can have significant consequences in situations where decisions are based on the received data. Thus, addressing QoS in routing protocols is vital, as it ensures that the network can prioritise and manage data delivery based on the specific needs of the applications, safeguarding the quality and usefulness of the transmitted information.

1.7 Conclusion

This chapter introduced the fundamental concepts underlying the IoT, emphasizing its enabling technologies, architectural layers, and wide range of application domains. Particular attention was given to the unique characteristics of IoT environments, such as constrained resources, dynamic topologies, and heterogeneous devices, that complicate network management and data communication.

The chapter also explored the pivotal role of routing in IoT systems, offering a classification of routing protocols tailored to different scenarios and design objectives. Key performance metrics were discussed, alongside the core challenges that arise in ensuring reliable, scalable, and energy-efficient routing.

By outlining these foundational elements, the chapter sets the stage for investigating more advanced and adaptive solutions. The next chapter introduces ML and DL techniques as promising tools to overcome the routing challenges in mobile and constrained IoT networks.

Chapter 2

Machine Learning and Deep Learning

2.1 Introduction

ML and DL have become indispensable tools for addressing complex decision-making tasks in modern communication systems, particularly in the context of the IoT [40]. These data-driven approaches enable systems to autonomously identify patterns and adapt their behaviour, making them especially effective in the decentralised, dynamic, and resource-constrained environments characteristic of mobile IoT networks.

This chapter presents the ML and DL algorithms that form the theoretical basis for the techniques applied in later chapters. The objective is to provide a foundational understanding of the models used in this research.

The first part of the chapter reviews traditional ML algorithms, including DT, SVM, and ANN. The second part introduces fundamental concepts of DL, with particular emphasis on CNN, highlighting their capacity to learn and generalise from complex, multidimensional data.

2.2 Machine learning

2.2.1 The idea behind machine learning: History

In traditional computer programming, the approach involves creating explicit rules (a sequence of instructions) within the code to guide the computer's actions based on input data. In this paradigm, the programmer is responsible for determining the rules that act upon the data, and the rules make up the bulk of the code. These predefined rules ultimately generate a desired answer, which can also be referred to as the output (see Figure 2.1.a). As an illustra-

tion, consider the creation of an algorithm to sort a set of numbers. The input data consists of these numbers; the desired answer (output) is a list of those numbers arranged in order. When addressing this task, multiple algorithms are available, each providing distinct sets of rules. The focus often revolves around identifying the most efficient one, aiming to minimize the usage of instructions and memory, or both, as needed. In certain instances, there are tasks for which we lack a specific set of rules or an algorithm, such as the task of distinguishing spam emails from legitimate ones. We can identify the input data as an email document, typically represented as a character file. Similarly, we can define the desired answer (output) as a binary decision, whether the message is spam (yes) or not (no). However, the challenge lies in finding the process that effectively transforms the input data into the desired output (answer). What qualifies as spam can vary over time and from person to person, making this task particularly complex. In many cases, we may not possess a deep understanding of a problem, but we can compensate for this by utilising vast datasets. For instance, when dealing with thousands of example messages, some known as spam, our objective is to teach a computer to automatically recognise spam by learning from the data. This entails extracting an algorithm or set of rules that can effectively distinguish spam from legitimate messages. While established algorithms exist for certain tasks like sorting numbers, many real-world applications lack predefined rules, and that's where example data comes into play, allowing us to automatically derive the necessary rules or algorithms for those tasks. Therefore, ML flips this traditional model by reversing the roles. Instead of explicitly defining the rules, you provide the computer with data and desired answers. Through ML algorithms, the computer autonomously discerns and learns the rules that can produce the desired outputs from the given data. This shift from rule-based programming to data-driven learning is the essence of ML (see figure 2.1.b).

2.2.2 Definition and scope

There are multiple formal definitions of ML in the literature [42] :

- In 1959, Arthur Samuel coined the term "Machine Learning", as "the field of study that gives computers the ability to learn without being explicitly programmed" [43].
- Later, Tom Mitchell (1997) provided a formal definition of learning, encapsulated in the following statement: "*A computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on T, as measured by P, improves with experience E*". This definition highlights the core

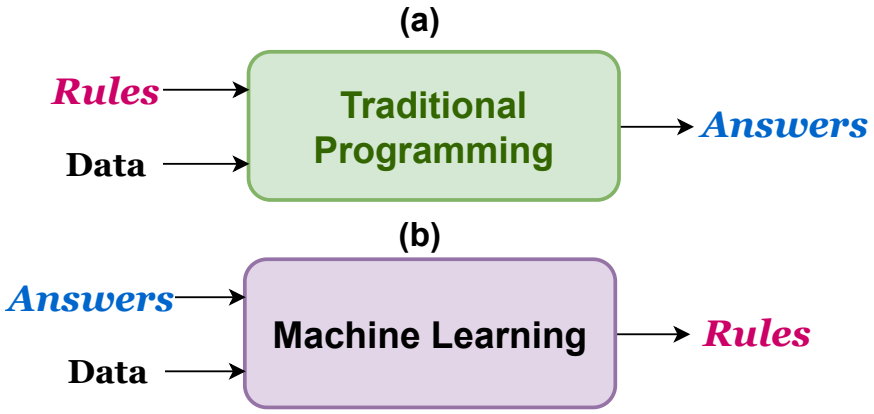

Figure 2.1: Traditional programming versus ML [41]

principle of ML, suggesting that a program can be considered to be learning when its performance on a specific task improves over time as it gains experience. The performance measure P quantitatively assesses the program's progress, indicating the extent of its evolving capability. [44].

- Ethem Alpaydin, in his textbook [44], further described ML as the field of '*programming computers to optimise a performance criterion using example data or past experience*'.

These definitions underscore the fundamental goal of ML: enabling computers to autonomously perform tasks and solve problems by adapting to new data and experiences. ML, a subfield of AI, focuses on developing algorithms that identify patterns and relationships in data, allowing systems to make informed predictions and decisions without explicit programming. Through continual exposure to data, these algorithms improve in accuracy and efficiency, enhancing a system's ability to handle complex and previously unseen situations effectively.

2.2.3 Learning paradigms

ML algorithms operate within distinct paradigms, each defined by the nature of the supervision they receive during training, as illustrated in Figure 2.2. Each paradigm adopts a unique approach to the learning process, shaping how models extract patterns from data. These paradigms [13] serve as foundational frameworks for developing intelligent systems. The primary categories include supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning, each of which will be discussed in detail.

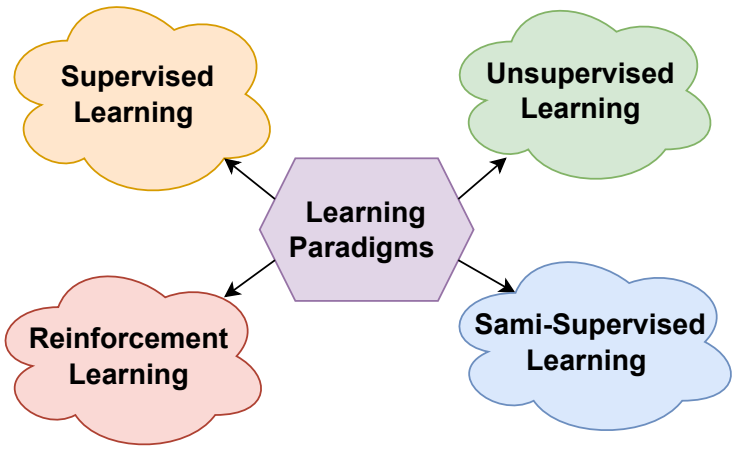

Figure 2.2: Machine learning paradigms

2.2.3.1 Supervised learning

In the context of supervised learning, having a labelled dataset is crucial. The labelled dataset consists of input-output pairs: the input variables (also known as features or attributes) and an output variable (the target). The input variables represent the characteristics or information associated with each instance, while the output variable represents the desired prediction, classification, or value to be determined. The process of assigning labels to the dataset is guided by a supervisor, from which the supervised learning appellation is inspired.

The primary goal of using a supervised learning technique is to find a deterministic function that maps any input to an output, enabling the construction of a dedicated model [45]. Once the classification model is generated, it can leverage this acquired knowledge to predict outcomes for new and previously unseen data instances. Supervised learning problems can be grouped into regression or classification problems:

- **Classification**: This involves predicting discrete class (category) labels based on input features (See Figure 2.3.a). For example, in email spam detection, the goal is to determine whether an email is "spam" or "not spam" based on its content and sender. Similarly, in image recognition, an algorithm identifies whether an image contains a "cat," "dog," or "car" by analysing pixel values.

- **Regression**: This focuses on predicting continuous numeric values (real numbers) using input features (See Figure 2.3.b). For instance, in house price prediction, the aim

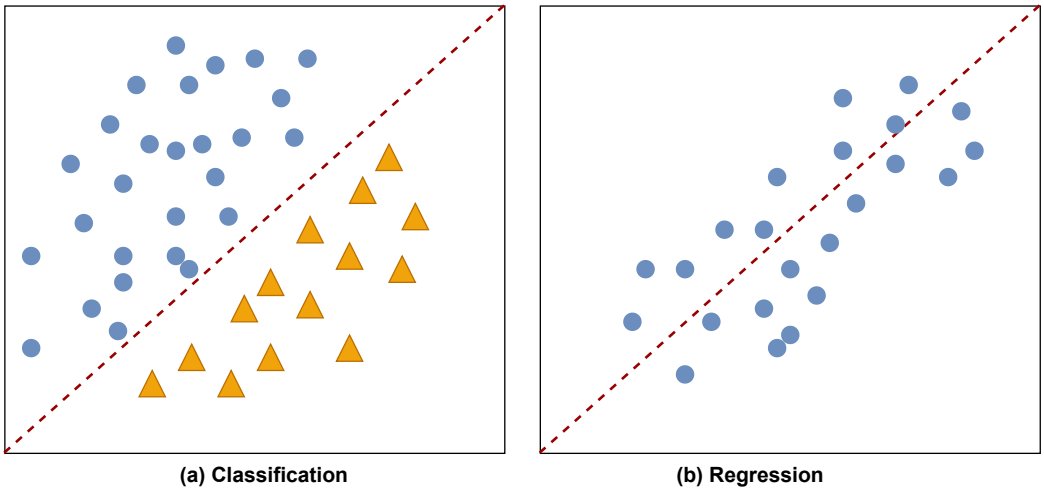

Figure 2.3: Supervised learning [45]

is to estimate the sale price of a house based on attributes like its square footage and number of bedrooms. In temperature forecasting, the algorithm predicts the temperature for the next day using historical data and time-related features. In both cases, the output is a continuous value that represents a quantity or measurement.

2.2.3.2 Unsupervised learning

Unsupervised learning involves training ML models on unlabeled datasets, where each data point lacks a specific label or class. The absence of labels means that predefined correct answers do not explicitly guide the algorithm during training. Instead, the algorithm aims to uncover inherent patterns and structures within the data. This contrasts with supervised learning, where models learn from labelled examples with clear target outputs.

The primary goal of unsupervised learning is to uncover meaningful structure and relationships within the data. By analysing the data's inherent characteristics, the algorithm seeks to identify similarities and groupings among data points. This process involves identifying patterns that might not be immediately obvious to a human observer.

A core aspect of unsupervised learning is measuring the similarity or dissimilarity between pairs of data objects. Various mathematical metrics quantify how alike or different two data points are. This similarity information is crucial for tasks such as clustering, where similar data points are grouped.

Density estimation is a fundamental concept in unsupervised learning. It involves estimating the probability density function that underlies the distribution of data points in the

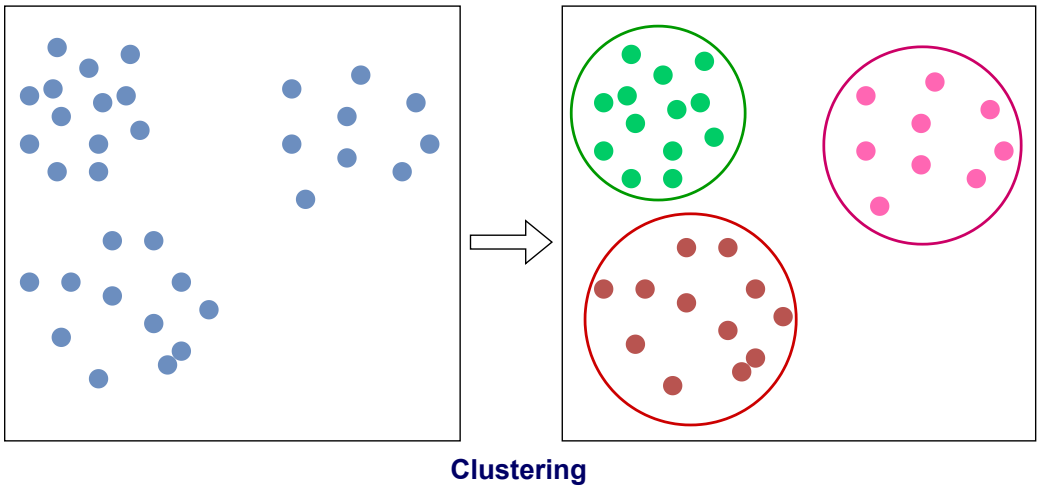

Figure 2.4: Unsupervised learning

feature space. By understanding the density of data points in different regions, the algorithm can gain insights into areas of high and low data concentration.

Unsupervised learning problems can be broadly categorised into two main groups: clustering and association problems.

- **Clustering:** involves grouping similar data points into clusters, where points within the same cluster are more similar to each other than to those in different clusters, as depicted in Figure 2.4. Clustering algorithms automatically identify these clusters based on computed similarities or distance measures. Common clustering algorithms include k-means, hierarchical clustering, and DBSCAN.

- **Association analysis:** focuses on identifying relationships or associations between different items in a dataset. A well-known example is market basket analysis, where the goal is to discover which items are frequently purchased together. These associations provide valuable insights for business strategies such as cross-selling and product recommendations.

2.2.3.3 Semi-supervised learning

Semi-supervised learning is a ML approach designed for scenarios where only a small portion of the data is labelled, while a substantial amount remains unlabeled. This methodology leverages both labelled and unlabeled data, bridging the gap between supervised and unsupervised learning [45].

By harnessing the inherent structures within unlabeled data through unsupervised techniques and incorporating the guidance of labelled examples, semi-supervised algorithms enhance learning efficiency. This approach enables models to generalise better, even with limited labelled samples. For instance, in document classification, unsupervised clustering can first group similar documents. Then, by assigning labels to a few labelled documents within each cluster, the model can propagate these labels to the remaining unlabeled documents in the same cluster, improving classification accuracy.

Semi-supervised learning is particularly useful in real-world applications where obtaining labelled data is costly or time-consuming, such as speech recognition, medical diagnosis, and web content classification. Various techniques, including self-training, co-training, and graph-based methods, are commonly used to exploit unlabeled data effectively [46].

2.2.3.4 Reinforcement learning

Unlike other ML paradigms, RL does not rely on a fixed dataset. Instead, it continuously collects data by interacting with its environment in an ongoing and adaptive learning process. RL trains agents to make sequential decisions to achieve a goal. At each discrete time step, the agent observes the current state, selects an action, and transitions to a new state based on environmental feedback. The objective is to learn a strategy, or policy, that maximises cumulative rewards over time, as illustrated in Figure 2.5 [47].

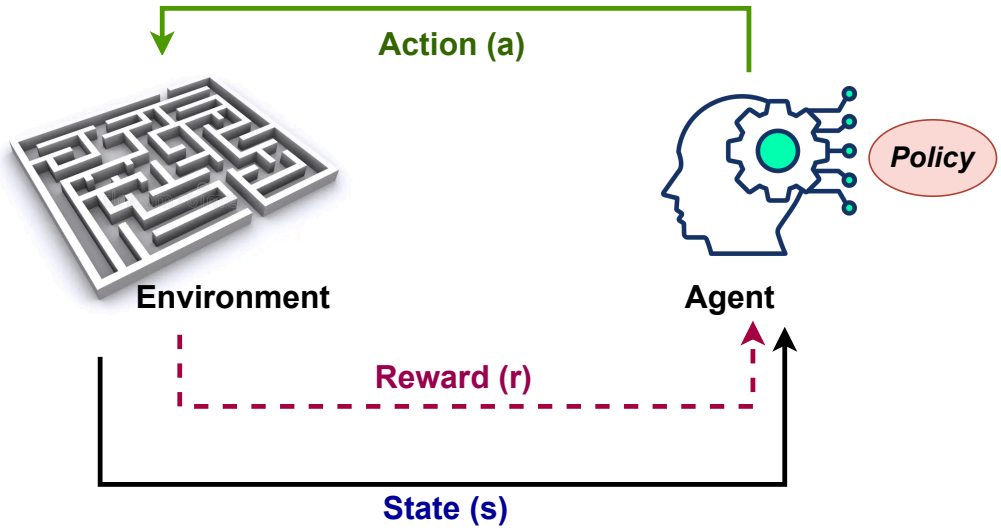

Figure 2.5: Reinforcement learning [48]

Here's a breakdown of the key components involved in RL and how they relate to each

other:

- **Agent**: The autonomous entity that learns by interacting with the environment to achieve a goal. The agent's decision-making process is guided by a policy it learns over time.
- **Environment**: The external system with which the agent interacts. It consists of a set of states, and the agent's actions lead to transitions between these states. The environment provides feedback in the form of rewards, indicating the effectiveness of the agent's actions.
- **State**: A representation of the current situation of the agent within the environment. It encapsulates the information the agent needs to make decisions. States can be fully observable, partially observable, or unobservable (Markov Decision Process (MDP) vs. Partially Observable MDP).
- **Action**: A decision made by the agent to influence the environment. The set of all possible actions constitutes the **action space**.
- **Reward**: A numerical value that quantifies the immediate benefit or cost of an action in a given state. The agent's objective is to maximise cumulative rewards over time.
- **Policy**: A strategy that defines the agent's behaviour by mapping states to actions. The goal is to learn an **optimal policy** that maximises the expected cumulative reward.

In the context of RL, the agent's learning process involves exploring the environment by taking actions and observing the resulting states and rewards. Through this exploration, the agent aims to understand the dynamics of the environment, learning which actions lead to more favorable outcomes. To achieve effective learning, various RL algorithms have been developed, including Q-learning, Deep Q Networks (DQN), Policy Gradient methods, and Actor-Critic methods, which guide the agent's learning process. These algorithms update the agent's policy based on the observed rewards and transitions between states, gradually improving its decision-making abilities.

The core challenge in RL is to strike a balance between **exploration**, which involves trying new actions to discover their effects, and **exploitation**, where the agent chooses actions that maximize rewards based on its current knowledge. This trade-off ensures that the agent continuously learns from experience while steadily improving its decision-making to maximize cumulative rewards [45].

2.2.4 Supervised ML workflow

Supervised learning involves a structured workflow that guides the development of predictive models using labelled data. As shown in Figure 2.6, the process consists of six main steps: problem definition, data collection, data preprocessing, model training, model validation, and model deployment. Each step plays a critical role in ensuring the effectiveness and generalizability of the final model. The following subsections provide a detailed explanation of each stage.

2.2.4.1 Problem definition

Defining the problem in a ML workflow is crucial. It involves a comprehensive and meticulous understanding of the challenge at hand. This step requires delving deeply into the specifics of the problem domain while considering the complexities, limitations, and potential consequences of the solution. Defining the problem entails setting clear and measurable goals, understanding what constitutes success, and aligning these objectives with the broader aims of the ML project. Furthermore, defining the problem isn't a static process; it often evolves through iterations, incorporating feedback, new information, or changing requirements. Thus, this pivotal stage lays the groundwork for the entire workflow, guiding subsequent steps such as data collection, preprocessing, model training, and deployment, ultimately steering the project toward its intended outcomes [49].

2.2.4.2 Data collection

ML techniques rely on unbiased, representative data to construct effective models for networking problems. Data collection is a crucial step because datasets that accurately represent a problem can vary not only between different problems but also across time. Moreover, the amount and quality of the data directly affect the model's effectiveness. Typically, data can be collected in two phases, offline and online. During offline collection, a wealth of historical data is amassed for training and testing models. In contrast, real-time network data collected during the online phase can serve as feedback or input for dynamically refining the model. Relevant offline data can be sourced from various repositories tailored to the networking problem at hand. For instance, repositories like Waikato Internet Traffic Storage (WITS), UCI Knowledge Discovery in Databases (KDD) Archive, and Measurement and Analysis on the WIDE Internet (MAWI) Working Group Traffic Archive exemplify such sources [13].

2.2.4.3 Data preprocessing

Raw data collected from various sources is often incomplete, inconsistent, or noisy. Before

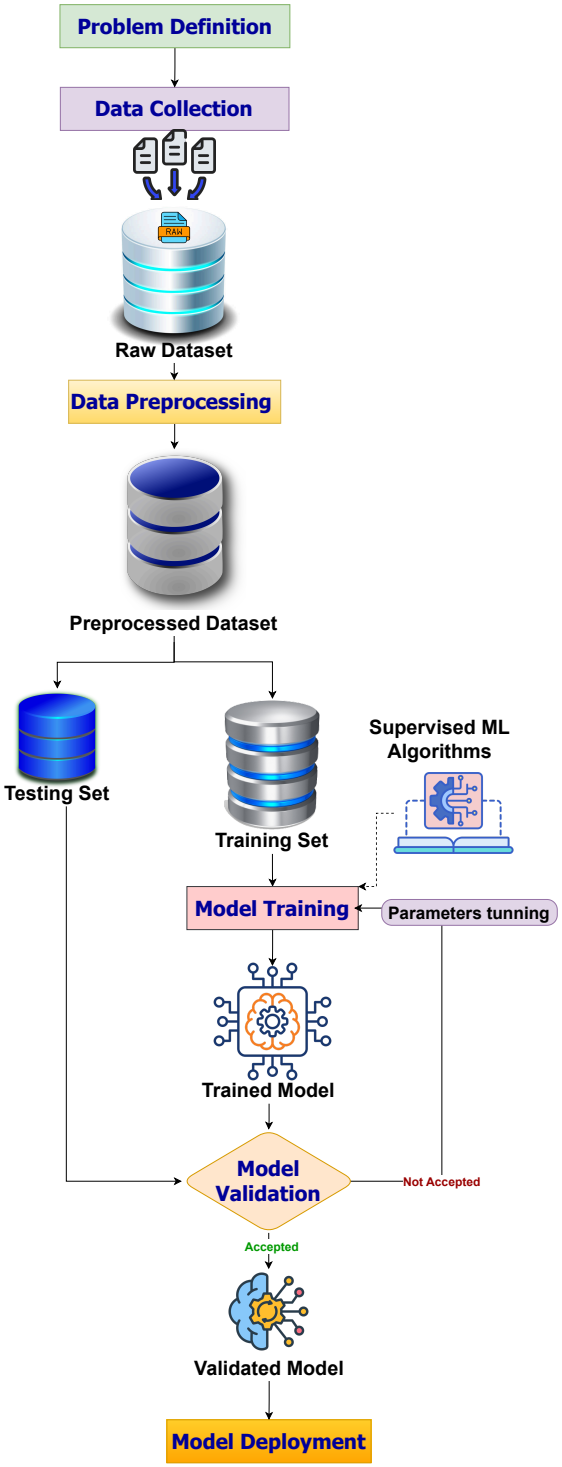

Figure 2.6: Supervised ML workflow

it can be effectively used for training ML models, it must undergo a preprocessing phase aimed at improving its quality and usability

The initial dataset may include a variety of imperfections, such as missing values, duplicate records, outliers, or discrepancies in formats and coding conventions. These issues, if left unaddressed, can significantly distort the learning process and reduce the accuracy and generalizability of the resulting model. Therefore, preprocessing is a crucial step that directly impacts the reliability of the entire ML workflow.

This phase typically involves several key tasks:

- Data cleaning: which removes or corrects erroneous entries and inconsistencies.
- Missing value imputation: where gaps in data are filled using statistical or model-based techniques.
- Outlier detection and handling: to mitigate the influence of anomalous values that could skew the model.
- Normalisation or standardisation: to bring data into a common scale without distorting differences in ranges.

For example, consider a dataset recording sales transactions that contains missing product entries for certain days. Ignoring these gaps can lead to misleading insights in time-series forecasting. Similarly, inconsistencies in how product categories are labelled (e.g., "TV", "television", and "T.V") may affect the grouping or summarisation of data.

By addressing such issues systematically, data preprocessing transforms raw input into a structured, reliable format. This clean and consistent dataset serves as the solid foundation upon which ML models can be accurately trained and evaluated [50, 51, 52].

2.2.4.4 Model training

After the data has been cleaned and prepared, it is divided into two sets: a training set and a testing set. Model training enables a ML algorithm to learn and determine optimal values for all the attributes present in the dataset. When we talk about an ML model, we refer to the outcome of this training process, which could be an actual model artefact or a decision function that encapsulates the learned patterns and relationships within the data. In supervised learning, the primary goal of model training is to create an accurate mathematical representation of how various attributes of the data relate to a specific target label. To achieve this, different supervised learning algorithms are employed. For instance,

DTs, which construct a flowchart-like structure to make decisions based on input features; SVM, focusing on finding the best boundary between different classes of data; Naïve Bayes, using probability and conditional independence assumptions; K-nearest neighbour (k-NN), which classifies data points based on their proximity to other known points; and ANN, designed to simulate the workings of the human brain in recognizing patterns [49].

2.2.4.5 Model validation

Model validation plays a critical role in assessing the accuracy and generalizability of a trained ML model. This step involves evaluating the model's performance on a separate dataset, commonly referred to as the testing or validation set, that was not used during training. The primary goal is to determine how well the model can make predictions on new, unseen data, thereby ensuring its readiness for deployment [53].

If the model achieves satisfactory performance, accurately predicting or classifying the instances in the test set, it is considered generalizable and suitable for real-world applications. However, if the performance falls below expectations, a model tuning phase is initiated. This involves adjusting hyperparameters, refining the model architecture, or revisiting the training data and techniques to improve accuracy and robustness [54].

For example, consider a recommendation system developed for an online streaming platform. The model is trained on user preferences, viewing history, and content metadata to suggest relevant shows or movies. After training, the model is validated using a separate dataset simulating real-world usage scenarios. If it consistently recommends content that aligns with user interests, the model can be deployed. Conversely, if the recommendations are inaccurate or inconsistent, further tuning is necessary. After refinement, the validation process is repeated to ensure the model meets the desired performance criteria before being released into production.

2.2.4.6 Model deployment

Once the model has demonstrated satisfactory performance during validation, the next step involves deploying it into a production environment where it can operate on real-time data and support decision-making processes. This transition is critical, as it marks the model's entry into practical use beyond controlled testing scenarios.

To ensure its continued effectiveness, the deployed model should be monitored consistently. Monitoring involves evaluating its predictions over time and verifying that it maintains acceptable performance as it encounters new and potentially evolving data patterns. Without such oversight, the model may suffer from performance degradation due to changes

in data distributions, a phenomenon known as concept drift.

To address this, mechanisms must be put in place for model updates and retraining. These mechanisms can involve periodic performance evaluations, feedback loops, and the integration of new data into the training pipeline. This allows the model to remain aligned with the operational environment and retain its predictive power over time.

In essence, deployment is not the final step in the ML workflow but rather the beginning of a continuous improvement cycle. Effective deployment is a lifecycle process that ensures the model remains adaptive, accurate, and aligned with evolving system requirements and user needs.

2.2.5 Supervised learning algorithms

In this study, we present several supervised learning algorithms that were applied during the model development phase. These algorithms represent distinct approaches to learning from labeled data and are used to construct models capable of making accurate predictions. The following subsections provide an overview of each algorithm utilised in our work.

2.2.5.1 The Decision Trees

A DT is a powerful machine-learning model representing decision-making processes in a structured, tree-like diagram [55]. It is primarily used for classification and regression tasks, as it systematically splits a dataset into subsets based on feature values, ultimately leading to a specific prediction or classification. DTs are known for their interpretability, as they mimic human decision-making by following a series of conditions or rules.

A. The structure of a DT

A DT consists of various components, as depicted in Figure 2.7, that work together to process data and make predictions:

- **Root node**: This is the starting point of the tree, representing the initial decision criterion. It contains the entire dataset before any splits occur.
- **Internal nodes (Test Nodes)**: These are decision points where the dataset is divided based on feature values. Each internal node corresponds to a specific attribute or feature evaluated using a condition.
- **Branches**: These are connections between nodes, representing possible outcomes of the conditions applied at internal nodes.

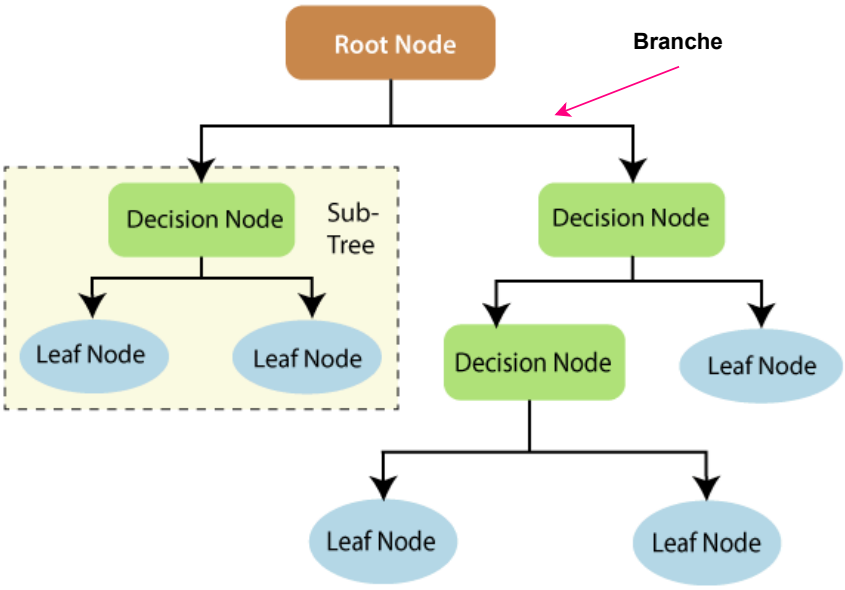

Figure 2.7: Basic components of a DT [56]

- **Leaf nodes (Terminal nodes)**: These nodes represent final outcomes. Each leaf contains a predicted class label (for classification tasks) or a continuous value (for regression tasks).

Each path from the root node to a leaf node forms a decision rule, determining how input features lead to specific predictions.

B. How DTs work

DTs operate by recursively partitioning data into smaller groups until a stopping criterion is met. This process follows a **top-down**, **recursive splitting** approach, often called the "divide and conquer" method. The primary goal is to select the most informative feature at each step to achieve the best separation of data points [57]. The key steps involved in constructing a DT are:

a) Feature selection: The algorithm evaluates all available features to determine which one provides the best split. This is typically done using impurity measures such as [55]:

- **Entropy and Information Gain (IG)**: Used in the Iterative Dichotomiser 3 (ID3) and C4.5 algorithms, entropy measures disorder in a dataset, while IG determines how much a feature reduces entropy.

- **Gini impurity**: Used in Classification and Regression Trees(CART), this measures the probability of misclassification if a sample were randomly classified based on feature distribution.

b) Splitting criteria: The dataset is divided into subsets based on the selected feature. In the case of categorical features, a branch is created for each category, whereas for numerical features, a threshold is chosen to split the data.

c) Recursive partitioning: The process repeats for each subset, treating them as new datasets until a stopping criterion is met. Common stopping criteria include:

- Maximum tree depth reached.
- Minimum number of samples per leaf node.
- No significant reduction in impurity.

d) Leaf Node assignment: Once the tree reaches a stopping condition, leaf nodes are assigned a final class label (classification) or a predicted value (regression). In classification, the majority class in the node is selected, while in regression, the average value is used.

C. DT algorithms

Several DT algorithms exist [58], each with different splitting criteria and methods:

- ID3: Uses entropy and IG to select features, only working with categorical data.
- C4.5: An extension of ID3 that handles both categorical and numerical data and includes pruning mechanisms.
- CART: Uses Gini impurity for classification and Mean Squared Error (MSE) for regression, allowing binary splits at each node.
- Chi-Square Automatic Interaction Detector (CHAID): Uses chi-square tests to determine feature splits, particularly useful in multi-way splits.

D. Overfitting and pruning

One of the major challenges in DTs is overfitting, where the model captures noise in the training data rather than general patterns. This happens when the tree becomes too deep and complex. To prevent overfitting, pruning is applied [55]:

- Pre-Pruning (Early Stopping): Limits tree growth by setting constraints such as maximum depth or minimum samples per node.
- Post-Pruning (Reduced Error Pruning, Cost Complexity Pruning): Involves training the full tree and then removing branches that do not improve performance on a validation set.

E. Advantages of DTs

- Interpretability: DTs are easy to understand and visualise, making them highly interpretable.
- Feature selection: Automatically selects the most important features.
- Handles mixed data Types: Works with both categorical and numerical data.
- Non-Parametric: No assumptions about data distribution are required.

F. Disadvantages of DTs

- Overfitting: Without pruning, trees can become overly complex.
- Instability: Small changes in data can lead to drastically different trees.
- Bias toward dominant features: Features with more categories or higher variance may dominate splits.

2.2.5.2 The Support vector machines

SVMs are robust supervised ML models initially developed for binary classification tasks, aiming to find an optimal hyperplane that best separates data points belonging to different classes in a high-dimensional feature space (Figure 2.8). This hyperplane, which in two dimensions appears as a line and in higher dimensions as a hyperplane, is positioned to maximise the margin between the nearest data points of each class, known as support vectors [59].

A. Key features and advantages

- Kernel trick: SVMs are kernel-based classifiers that leverage the kernel trick to handle non-linearly separable data. By transforming the original input space into a higher-dimensional space where data points become separable by a hyperplane, SVMs can

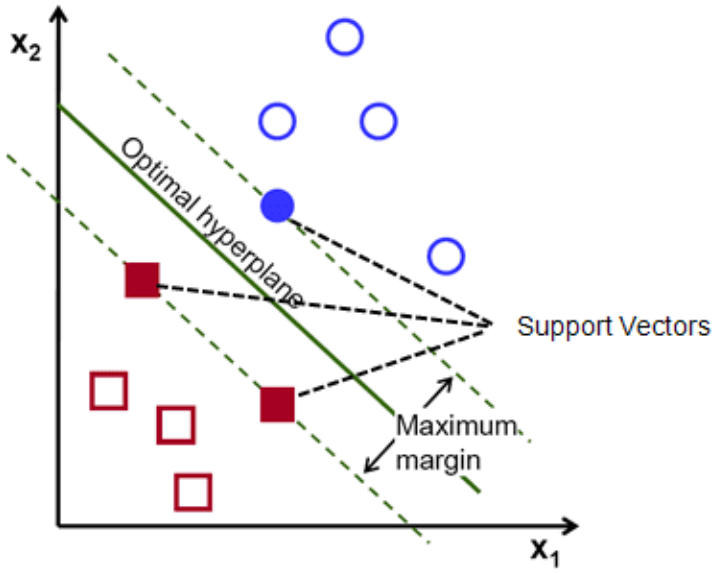

Figure 2.8: Basic components of the SVM [60]

effectively model complex relationships in data that wouldn't be linearly separable otherwise.

- Margin maximisation: The core principle of SVMs is to maximise the margin, which enhances the model's ability to generalise to unseen data and reduces the risk of overfitting. This margin is controlled by a parameter 'C' that balances the trade-off between maximising the margin and minimising classification errors.
- Types of Kernels: SVMs support different kernel functions, each suited for different types of data distributions:
  - **Radial Basis Function (RBF) Kernel**: Most commonly used due to its versatility in capturing complex relationships. It's governed by parameters 'C' and 'gamma', where 'C' controls the soft margin and 'gamma' influences the shape of the decision boundary.
  - **Polynomial kernel**: Suitable for capturing polynomial relationships in data, with parameters for degree and constant term adjustments.
  - **Sigmoid kernel**: Used for problems where decision boundaries have a sigmoid shape, controlled by parameters that determine slope and intercept.
- **Multi-class classification**: Initially designed for binary classification, SVMs can be extended to handle multi-class problems through strategies like 'one-vs-rest' or 'one-

vs-all'. Here, separate binary classifiers are trained for each class, and the class with the highest decision score across all classifiers is chosen as the final prediction.

B. Practical considerations

- **Model tuning**: Selecting the appropriate kernel and tuning its parameters is crucial for optimising SVM performance. Techniques like cross-validation help identify the best combination of parameters that maximise model accuracy and generalisation.
- **Scalability**: SVMs are effective in high-dimensional spaces and are memory efficient because they primarily use a subset of training points (support vectors) in the decision function.
- **Applications**: SVMs find applications in diverse fields such as bioinformatics, image recognition, text classification, and financial forecasting, where data may exhibit complex non-linear relationships that require robust classification or regression models.

2.2.5.3 The Artificial Neural Networks

A. The Structure of an ANN

An ANN is a computational model inspired by the biological neural networks found in the human brain. It is widely used in ML for tasks such as pattern recognition, classification, regression, and decision-making across various domains, including computer vision, natural language processing, and medical diagnosis. An ANN consists of interconnected processing units called neurons, which are structured into three main layers (see Figure 2.9):

- **Input layer**: The first layer receives raw input data, where each neuron corresponds to a specific feature or input variable. The number of neurons in this layer matches the number of input features in the dataset.
- **Hidden layers**: These intermediate layers process the input data through weighted connections, capturing complex relationships and feature hierarchies. The number of hidden layers and neurons in each layer can vary depending on the complexity of the problem. Deeper networks with multiple hidden layers are referred to as Deep Neural Networks (DNNs).
- **Output layer:** The final layer produces the network's predictions. The number of output neurons depends on the task: a single node for binary classification, multiple nodes for multi-class classification, or continuous output for regression tasks.

Each neuron in a neural network receives inputs, processes them using an activation function, and passes the output to the next layer. The network's effectiveness depends on how well the hidden layers extract meaningful patterns from the input data.

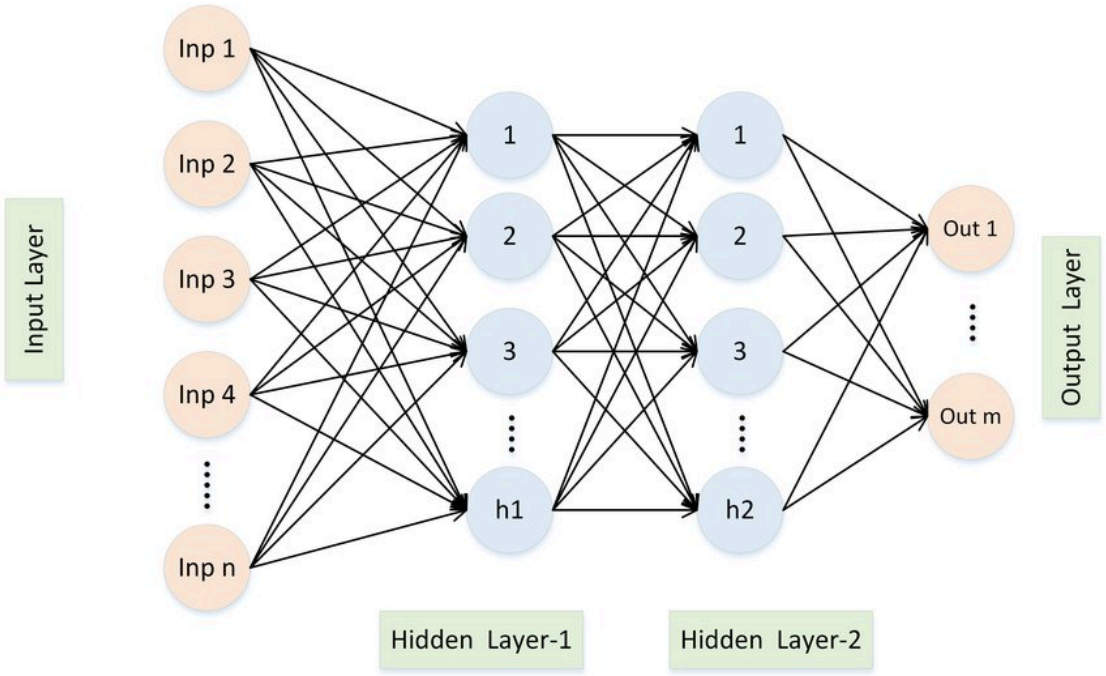

Figure 2.9: Architecture of an ANN [61]

B. The Functioning of a neuron

At the core of an ANN is the artificial neuron, a computational unit that processes input data and generates an output (Figure 2.10). The operation of a neuron follows these steps:

a) **Weighted Summation:** Each neuron receives inputs x_i from the previous layer and assigns a corresponding weight w_i to each input. The neuron computes the weighted sum as:

$$S = \sum_{i=1}^{n} (x_i \cdot w_i) + b \tag{2.1}$$

where b is the bias term, which helps shift the activation function to improve learning stability.

b) **Activation function:** The weighted sum S is passed through an activation function, introducing non-linearity to help the network learn complex patterns. Common

activation functions include:

- **Sigmoid**: used for binary classification but prone to vanishing gradients in deep networks.
$$\sigma(S) = \frac{1}{1 + e^{-S}} \tag{2.2}$$
- **Tanh (Hyperbolic Tangent)**: scales outputs between -1 and 1, improving training over sigmoid.
- **ReLU (Rectified Linear Unit)**: popular for DL due to its efficiency in handling large-scale data and mitigating the vanishing gradient problem.
- **Softmax**: Often used in the output layer for multi-class classification, transforming outputs into probabilities.

These activation functions allow neural networks to model complex decision boundaries and approximate any continuous function, a property known as the universal approximation theorem.

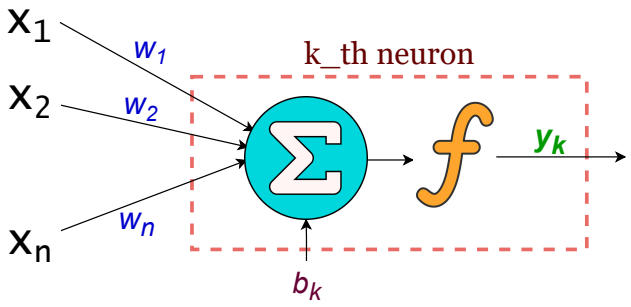

Figure 2.10: Model of a single artificial neuron [59]

C. Training process of ANNs

Training an ANN involves adjusting the weights and biases through a process known as backpropagation, which minimises the error between predicted and actual outputs. The training process consists of the following steps:

a) **Forward pass:** where input data is propagated through the network, layer by layer. Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer. This process continues until the output layer generates the final prediction.

b) **Error computation:** This is performed by comparing the network's predicted output to the actual target values using a suitable loss function. For regression tasks, the MSE is commonly used, while classification problems often employ Cross-Entropy loss. The computed error serves as the basis for adjusting the network's weights to improve accuracy.

c) **Backpropagation:** To update the network's parameters, backpropagation is employed. This algorithm propagates the error backwards through the network using the chain rule of calculus. Gradients of the loss function concerning each weight are computed, allowing the model to determine how much each parameter contributed to the error.

d) **Gradient descent optimisation:** Once gradients are obtained, Gradient descent optimisation is used to adjust the weights in the direction that minimises the loss. Several optimisation techniques exist, including *Stochastic Gradient Descent (SGD)*, which updates weights after processing each sample, and the *Adam Optimiser*, an adaptive method that combines momentum and RMSProp for efficient learning in deep networks.

e) **Iteration and convergence:** Finally, Iteration and convergence take place. The forward pass, error computation, backpropagation, and weight updates are repeated over multiple epochs. This iterative process continues until the model achieves a satisfactory level of performance, as indicated by stable loss values and improved accuracy on the validation data.

Regularisation techniques, such as dropout, L1/L2 weight decay, and batch normalisation, help prevent overfitting and improve generalisation to unseen data.

2.3 Deep Learning

2.3.1 Introduction to deep learning

DL is a branch of AI and a modern evolution of ML, distinguished by its ability to learn complex and abstract data representations through deep architectures automatically. The term DL was popularised in 2006 by Geoffrey Hinton and his colleagues [62], who demonstrated how multiple layers of neural networks could be trained effectively using unsupervised

pretraining strategies. Since then, DL has gained significant traction due to its exceptional performance in tasks involving large and complex datasets.

DL is built upon neural networks that consist of layers of interconnected neurons. These networks learn to extract hierarchical features from input data, making them suitable for tasks where traditional ML techniques fall short, such as image classification, speech recognition, and natural language processing. In recent years, DL has also shown promise in fields such as healthcare, autonomous vehicles, and network systems, including intelligent routing decisions in dynamic environments [63].

2.3.1.1 Artificial neural networks

The concept of ANNs was first inspired by the biological neural networks found in the human brain. Mathematically, an ANN consists of layers of neurons that apply nonlinear transformations to input features. A basic ANN typically includes:

- **Input layer** : receives the raw data.
- **Hidden layer(s)** : performs computations and feature transformations.
- **Output layer**: produces the final result, such as a classification or prediction.

Each neuron in the network computes a weighted sum of its inputs and applies an activation function such as the sigmoid or ReLU. ANNs can learn from data through iterative training using algorithms like backpropagation combined with gradient descent optimisation.

Although effective for small-scale problems, traditional ANNs, also called **shallow networks**, typically involve only one or two hidden layers, which limits their capacity to model high-level abstractions in data.

2.3.1.1 Deep neural networks

DNNs extend traditional ANNs by incorporating many hidden layers between the input and output layers. This depth allows DNNs to model complex and abstract patterns by learning hierarchical feature representations, effectively addressing the limitations of shallow networks.

As illustrated in Figure 2.11, shallow ANNs usually consist of only one or two hidden layers, whereas DNNs stack multiple layers to form deep architectures capable of extracting increasingly abstract features from data.

The general architecture of a DNN can be described as follows:

- **Input layer**: Converts raw input into a vectorised form, assigning weights to inputs.

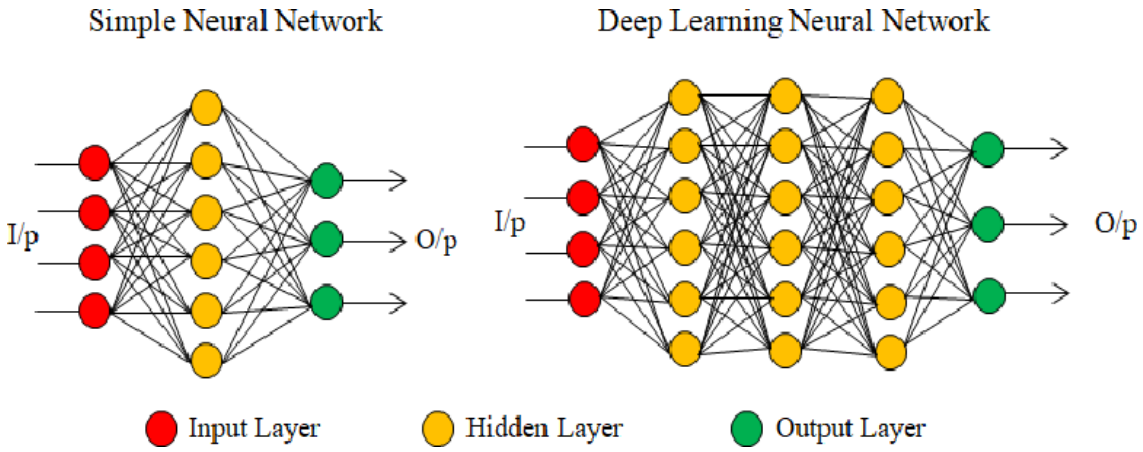

Figure 2.11: Difference between Shallow ANN and Deep ANN [64]

- **Multiple hidden layers**: Each layer processes its input using operations such as filtering and feature extraction, then passes its output to the next layer.
- **Output layer**: Aggregates the final representation and produces a prediction or decision.

One of the most basic forms of DNNs is the Multi-Layer Perceptron (MLP), a fully connected feedforward network where each neuron in one layer is connected to every neuron in the next. Like shallow ANNs, DNNs are trained using backpropagation, but they benefit from enhanced training techniques that make deep architectures viable and scalable.

DNNs offer the following key advantages:

- Automated feature learning: They reduce the reliance on manual feature engineering.
- Hierarchical representation: Each layer learns more abstract features based on the output of the previous layer.
- Scalability: With increased data and computing power, DNNs can learn more robust models.

Since 2006, the development of more efficient training algorithms, the availability of large-scale datasets, and advancements in hardware (e.g., GPUs) have led to a rapid expansion of DL research and applications.

2.3.2 Advantages and breakthroughs of DL

Several factors have contributed to the success and widespread adoption of DL:

- Computational power: The rise of GPUs and distributed computing has dramatically reduced training time.
- Big data availability: DL thrives on large datasets, and the exponential growth of data has supported its expansion.
- Improved training techniques: Innovations such as ReLU activation functions, dropout regularisation, and advanced optimisers (e.g., Adam) have enhanced model accuracy and generalisation.
- End-to-end learning: DL models can be trained directly from raw input to output, eliminating the need for handcrafted features.

DL methods have demonstrated significant success in diverse domains, including but not limited to: image recognition (e.g., object detection, facial recognition), speech and audio processing, natural language understanding, autonomous driving, and Medical imaging analysis.

These advancements have also led to new exploration in areas such as network traffic prediction and routing optimisation.

2.3.3 Convolutional neural network

Among the various DL architectures, CNNs are particularly well-known for their success in visual data processing tasks. CNNs are specialised neural networks that exploit spatial hierarchies in data through the use of convolutional layers, pooling layers, and fully connected layers.

CNNs were first introduced by Yann LeCun in 1989 [65] for document recognition and have since become a cornerstone in the field of AI, powering advances in computer vision, natural language processing, medical imaging, and more. What distinguishes CNNs is their ability to automatically learn hierarchical features directly from raw input data, eliminating the need for manual feature engineering. This is achieved by progressively extracting and abstracting patterns across multiple layers, each operating at a different level of granularity.

CNNs are a specialised class of DL models designed to process data with spatial or temporal hierarchies, such as images, audio signals, and time-series data. Introduced by Yann LeCun in 1989 for document recognition tasks, CNNs have since become a cornerstone in the field of AI, driving advancements in computer vision, natural language processing, medical imaging, and more. Their popularity stems from their ability to **automatically learn**

hierarchical features directly from raw data without the need for manual feature engineering. This is achieved through a structured composition of multiple layers, convolution, activation (ReLU), pooling, flattening, and fully connected layers, that progressively extract and abstract patterns at different levels of granularity, as illustrated in Figure 2.12.

At the heart of CNNs lies the **convolutional layer**, which performs a mathematical operation called convolution. This layer uses multiple learnable filters (also called kernels) that slide over the input data to detect localised features such as edges, textures, or specific patterns. Each filter produces a feature map by computing the dot product between the filter and the overlapping regions of the input. This operation leverages **weight sharing**, where the same filter is applied across the entire input, reducing the number of parameters and making the model more efficient. Additionally, CNNs exploit **sparse interactions**, meaning that each output value depends only on a small region of the input, allowing the network to capture local dependencies. The output of a convolutional layer is usually passed through a nonlinear activation function, commonly ReLU, to introduce nonlinearity and enable the network to learn complex representations.

Following convolution, CNNs often employ **pooling layers**, which reduce the spatial dimensions of the feature maps while retaining the most important information. This process, known as down-sampling or sub-sampling, serves two key purposes: it reduces **computational complexity** and increases the model's robustness to small variations or distortions in the input. Common pooling operations include **max pooling**, which selects the maximum value in each local region, and average pooling, which computes the average. These operations help the network focus on the most dominant features, enabling shift invariance and improving generalisation in unseen data.

Before feeding the extracted features into the final decision-making layers, CNNs apply a **flattening** operation, which transforms the two-dimensional feature maps into a one-dimensional vector. This step is necessary because fully connected layers expect vector input. Flattening effectively bridges the convolutional/pooling part of the network with the classification (dense) layers.

Towards the end of a CNN architecture, the extracted features are typically fed into one or more **fully connected (dense) layers**, where each neuron is connected to every neuron in the previous layer. These layers act as the decision-making component of the network. By combining high-level features extracted by the earlier convolutional and pooling layers, the fully connected layers learn to perform classification or regression tasks. For example, in an image classification task, the final fully connected layer might produce a probability

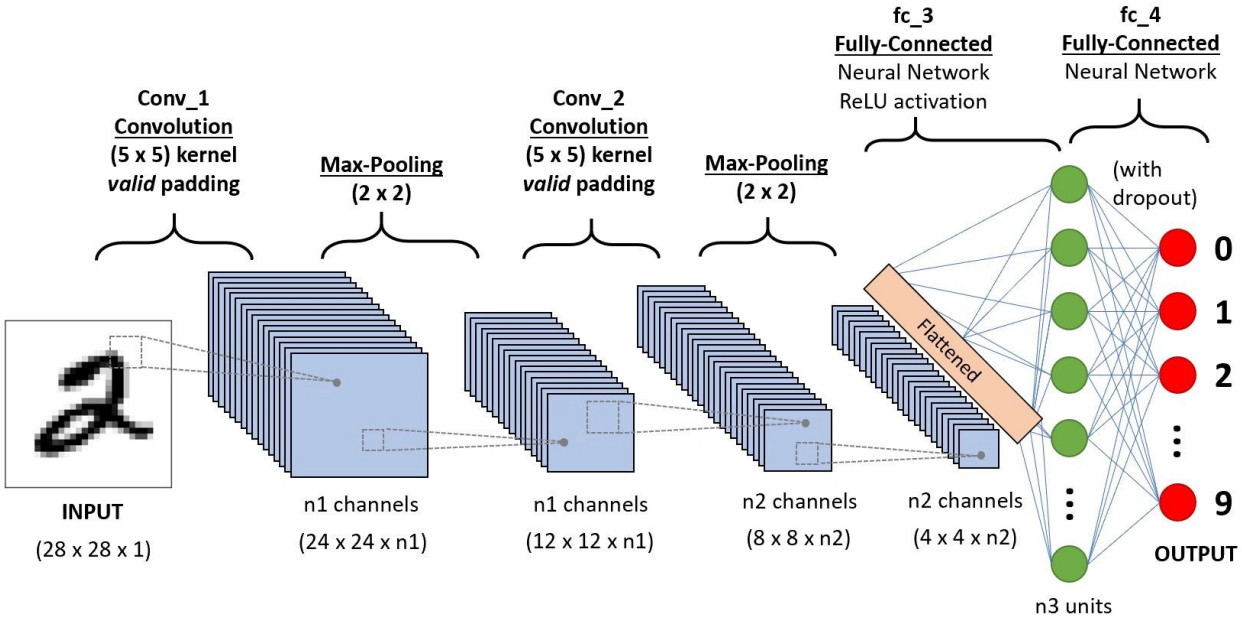

Figure 2.12: Illustration of a typical CNN architecture [66]

distribution over the set of possible categories.

To further improve generalisation and prevent overfitting, CNN architectures often incorporate regularisation techniques such as **dropout**, where a subset of neurons is randomly disabled during training. This forces the network to learn redundant representations and prevents reliance on any single feature. **Batch normalisation** is another widely used technique that stabilises and accelerates training by normalising the input to each layer.

2.4 Conclusion

This chapter introduced the key ML and DL techniques relevant to this thesis. It reviewed classical ML algorithms, including DTs, SVMs, and ANNs, followed by an overview of DL principles with a particular focus on CNNs.

These techniques form the methodological foundation for the models explored later in this work. Their specific application will be addressed in the dedicated contribution chapters.

The following chapter presents a review of related works, highlighting existing approaches in the field and identifying research gaps that motivate the proposed solutions.

Chapter 3

Related Works

3.1 Introduction

The growing complexity of mobile and dynamic networks, such as IoT, MANETs, and VANETs, has exposed the limitations of traditional routing protocols in handling topology changes, unstable links, and energy constraints. To overcome these challenges, intelligent routing strategies powered by ML and DL have gained prominence for their adaptability and efficiency.

This chapter draws upon insights from our previous surveys [67, 68] and presents a comprehensive review of ML- and DL-based routing techniques. It begins with early RL approaches, then examines classical supervised methods such as SVMs, DTs, and Random Forests. The discussion then extends to recent developments involving Deep Reinforcement Learning (DRL), CNNs, Recurrent Neural Networks (RNNs), Artificial Neural Networks (ANNs, and Graph Neural Networks (GNNs). Finally, hybrid models and cross-layer designs are considered, laying the groundwork for the research contributions presented in the subsequent chapters.

3.2 ML-based routing

3.2.1 RL-based routing

The use of RL in network traffic routing dates back to the early 1990s, notably with the development of Q-routing by Boyan and Littman [69], which directly applied the Q-learning algorithm. In Q-routing, routers estimate delivery times through Q-values, allowing them to adjust routes dynamically without requiring prior information about the network topology.

This method outperformed traditional shortest-path approaches under heavy traffic, although frequent updates to routing policies could lead to congestion.

Following Q-routing, numerous RL-based routing algorithms were proposed as extensions [70]. For example, Predictive Q-routing (PQ-routing) [71] aimed to accelerate convergence by leveraging previously estimated delivery times for neighbouring nodes. Similarly, Dual Reinforcement Q-routing (DRQ-Routing) [72] incorporated both upstream and downstream feedback to enhance convergence rates, with only a slight increase in communication overhead. In the context of multicast routing, Q-MAP [73] tailored Q-learning to construct efficient multicast trees in MANETs, illustrating the flexibility of RL across different network types.

RL techniques have also been adapted to support mobility and dynamic topologies. For instance, Mobility-Aware RL-based Routing [74] updated Q-values based on mobility behaviour, encouraging exploration of new paths in changing environments. In another example, RL-based AODV [75] enhanced the conventional AODV protocol [76] by employing Q-values for next-hop decisions, allowing it to respond to network state variations using neural networks.

Energy conservation has become another central theme in RL-based routing. RL-based Constrained Flooding (RLCF) [77] reduced transmission overhead in WSNs through a flooding mechanism constrained by RL, thus saving energy without control messages. Adaptive Routing (AdaR) [78] utilised multiple metrics, such as delay, energy, and link quality, to guide routing choices. RL for Green Routing (RLGR) [79] selected forwarding nodes based on remaining energy levels, helping to extend the overall lifetime of the network.

In scenarios requiring QoS, RL has also been applied. RL-QRP [80] enabled timely and reliable packet delivery by focusing on QoS indicators and dropping packets that failed to meet preset thresholds, making it applicable to domains like healthcare, where timing is critical.

Additionally, hybrid approaches have emerged, such as RLGAMAN [81], which integrates RL with genetic algorithms to discover multiple QoS-compliant routes. CQLAODV [82] combined RL and Bayesian networks to select next-hop nodes based on probability tables, thereby enhancing the flexibility and accuracy of routing decisions.

Altogether, these works reflect the progression of RL in routing, from its foundational techniques to more advanced, context-aware, and hybrid models, demonstrating its growing impact on a wide variety of network environments.

More recent studies have extended RL to suit complex and specialised environments. Routray and Sharmila [83] modelled MANET routing as a MDP, enabling nodes to au-

tonomously adjust to topology changes. Their Q-learning-based method improved adaptability, reduced delay, and enhanced packet delivery compared to traditional routing schemes.

In underwater sensor networks, QELAR [84] applied Q-learning to optimise forwarding decisions based on residual energy and communication delay. Designed for energy-constrained and delay-tolerant settings, QELAR achieved notable gains in energy efficiency and delivery ratio.

Warp-5 [85] proposed a cross-layer, context-aware routing protocol that incorporated statistical learning of congestion and environmental noise. By utilizing recent network history, Warp-5 dynamically adjusted routing decisions and outperformed both Q-routing and AODV in noisy wireless conditions.

Lastly, in the context of self-organising 5G networks, Murudkar and Gitlin [86] introduced a learning-based routing approach that integrated supervised learning with RL inspiration. Their system selected optimal paths based on real-time link capacity and quality metrics, achieving better resource usage and reduced latency compared to heuristic methods.

3.2.2 Supervised learning-based routing

3.2.2.1 DT-based routing

DT classifiers have proven effective for routing decisions in mobile networks due to their simplicity and low computational overhead. In [87], the authors propose the DT-based Routing Protocol (DTRP), which enhances route stability in MANETs by analysing features such as link expiration time, node velocity, and trip duration. The trained DT model guides the selection of the most reliable next-hop during the route discovery phase, resulting in reduced link failures and improved packet delivery when compared to traditional reactive protocols.

Building on the use of DTs in dynamic environments, DT-VAR [88] introduces a trust-aware routing framework for fog-assisted VANETs. This method employs a DT classifier trained on features like expected link compatibility time and node trust levels to predict stable and trustworthy forwarding paths. The approach achieves a classification accuracy of up to 99% and significantly improves the delivery ratio by 12% over conventional protocols such as AOMDV and TCSR. These results highlight the potential of lightweight DT models for supporting real-time, mobility-aware routing in vehicular networks.

3.2.2.2 SVM-based routing

SVMs have been employed in routing protocols for their ability to handle complex classification tasks with high accuracy, particularly in dynamic and resource-constrained envi-

ronments. In [89], an SVM model is used to evaluate node trustworthiness based on metrics such as mobility, packet drop ratio, and forwarding behaviour. When the primary route becomes unreliable, the system proactively identifies alternative paths, resulting in improved throughput and reduced end-to-end delay in simulation scenarios.

Expanding on the use of SVM for decision-making, [90] applies the model in WSNs to predict energy-efficient paths. Here, the SVM is trained using features such as residual energy and distance to the sink, allowing the system to avoid energy-depleted nodes. This leads to significant improvements in network lifetime, load distribution, and packet delivery when compared to conventional routing protocols.

In a similar vein, [91] leverages SVM for relay node selection in high-mobility vehicular networks. The classifier considers parameters like node speed, link quality, and neighbourhood density to identify stable relay candidates. The approach enhances delivery performance and reduces latency, with minimal computational cost, making it well-suited for the fast-changing conditions of vehicular environments.

3.2.2.3 Hybrid approaches

To enhance routing performance, several studies have combined multiple machine learning methods to support more effective decision-making. These approaches integrate different models to improve routing accuracy, adaptability, and efficiency in diverse network environments.

For instance, although not a routing protocol in itself, the study in [92] evaluates multiple classifiers, J48, Naïve Bayes, Logistic Regression, and Random Forest, to classify and filter mobile client requests before reaching the cloud. Among these, Random Forest achieved the highest accuracy (86.36%), demonstrating its potential to reduce upstream traffic and indirectly enhance routing efficiency in mobile edge computing contexts.

In a related effort, [93] proposes a supervised learning-based routing mechanism tailored for smart airport environments. By using features such as device density, node connectivity, and congestion level, the system classifies the most suitable message forwarding paths. The results show significant improvements in latency and congestion mitigation over conventional routing protocols.

In the vehicular domain, [94] introduces the GraTree protocol, which employs Gradient Boosted Decision Trees (GBDT) to predict reliable forwarding routes. By considering factors like link quality, vehicle velocity, and hop count, GraTree outperforms geographic and multimetric routing schemes in urban VANET simulations, underscoring the effectiveness of advanced learning models in high-mobility scenarios.

Beyond ensemble-based methods, [95] proposes a multi-layer predictive model for routing in opportunistic IoT networks. A logistic regression classifier first estimates the likelihood of successful message delivery, followed by a neural network that refines the routing decision. Tested on real-world contact traces, this approach improves delivery ratios and reduces transmission overhead, particularly under delay-prone conditions.

3.3 DL-based routing

3.3.1 RNN-based routing

RNNs have emerged as effective tools for modelling sequential and temporal data in network environments. Their ability to learn from historical patterns makes them particularly well-suited for routing scenarios where past network behaviour influences current forwarding decisions. The following studies demonstrate the application of RNNs in enhancing routing performance across various types of dynamic and resource-constrained networks.

In wireless mesh networks (WMNs), Eyobu et al. [96] utilise Long Short-Term Memory (LSTM) networks to predict efficient paths through a 2-hop data collection technique, emphasising QoS parameters such as packet delivery ratio and throughput. Their method reduces errors and delays compared to traditional protocols like AODV. In a similar context, Nagalingayya et al. [97] propose an RNN-based solution for wireless multimedia sensor networks (WMSNs), aiming to enhance energy efficiency by learning from previous network states. This strategy improves data delivery and extends network lifespan.

Within IoT scenarios, Sumathi et al. [98] present the NEWTR model, which leverages RNNs to select next-hop nodes based on current traffic trends. By using a multipath routing approach, the method improves both fault tolerance and energy usage. In the domain of Software-Defined Networking (SDN), Modi et al. [99] apply RNNs to predict traffic behaviour in data centre networks, allowing dynamic path updates that reduce latency and optimize bandwidth allocation.

In MANETs, Pal et al. [100] introduce an RNN-based multipath routing mechanism to improve load distribution. Their method predicts traffic levels and adjusts routes accordingly, minimising congestion and enhancing performance. Addressing security in WSNs, Venkatesh et al. [101] design a dual encoding RNN to assess node trustworthiness. This classification approach reduces routing overhead and delay while improving energy efficiency, offering a secure data transmission strategy.

Collectively, these contributions illustrate the effectiveness of RNNs in handling the vari-

ability of modern networks, supporting improvements in energy use, performance, and security across various deployment scenarios.

3.3.2 CNN-based routing

In addition to RNNs, CNNs have been investigated for routing tasks by exploiting spatial data patterns. Modi et al. [102] present a CNN-based approach for software-defined data centre networks (SDDCNs), where the model uses network topology and traffic characteristics to predict efficient paths. This results in reduced latency and improved throughput. In another application, Darwassh et al. [103] propose a hybrid BA-CNN model for emergency vehicle routing in smart city environments. The Bat Algorithm is employed to optimise CNN parameters, enabling the system to process real-time traffic inputs and enhance routing accuracy and response time. These studies highlight the applicability of CNNs to complex routing scenarios in both urban and data centre settings.

3.3.3 GNN-based routing

GNNs offer a powerful means of modelling network topologies by incorporating structural information inherent to graph-based systems. Their ability to learn intricate relationships between nodes makes them highly suitable for dynamic routing tasks. Geyer et al. [104] utilise GNNs to develop adaptive distributed routing protocols that analyze both topology and node-level data, achieving notable reductions in packet loss and latency. Similarly, Yan et al. [105] apply GNNs to optimise multipath routing in SDNs, enhancing flowlet-level decision-making, load balancing, and congestion mitigation.

Further advancing this line of research, Wei et al. [106] introduce a real-time routing framework that dynamically adapts to changing network states using a GNN-based representation of topology, resulting in improved path selection and lower latency. Rusek et al. [107] present RouteNet, a GNN-powered model for SDNs that learns the relationship between network conditions and performance metrics, demonstrating superior results compared to traditional routing techniques. Collectively, these works affirm the strength of GNNs in addressing complex routing challenges across modern network infrastructures.

3.3.4 DNN-based routing

DNNs have been applied to address complex routing challenges in diverse and dynamic network settings. Liu et al. [108] propose a DNN-based routing mechanism for Aeronautical

Ad Hoc Networks (AANETs), where the model utilises local geographic information to determine optimal next-hop nodes. By learning the relationship between local observations and the minimum delay to a destination, the approach achieves reductions in end-to-end delay, enhances path capacity, and prolongs route lifetime, outperforming conventional routing protocols.

Similarly, Reis et al. [109] introduce a DNN model designed to alleviate network congestion by predicting efficient routing paths using historical traffic data. To overcome the limitations of computationally intensive methods like Mixed-Integer Linear Programming (MILP), their solution trains a neural network on datasets containing historical demand and optimal route information. The resulting model enables fast routing decisions that closely approximate optimal outcomes, contributing to lower latency and improved traffic flow.

These contributions highlight the effectiveness of DNNs in supporting real-time, adaptive routing, demonstrating their value in managing complex network dynamics.

3.3.5 Hybrid approaches

Numerous studies have proposed hybrid models that combine DRL with other DL techniques to enhance routing performance in dynamic network environments.

A significant area of focus has been the integration of GNNs with DRL. Swaminathan et al. [110] introduced "GraphNET", a framework that combines GNNs and DRL to improve routing in SDNs. In this approach, GNNs learn node embeddings from simulated data, which are then used by a DRL agent to make adaptive routing decisions, resulting in increased throughput and lower latency.

Expanding on this concept, Chen et al. [111] propose a two-layer model in which GNNs capture spatial relationships among nodes, while the DRL agent derives optimal routing policies through interaction with the network. This architecture offers improved scalability and performance compared to conventional routing techniques.

Hope and Yoneki [112] present "GDDR", a hybrid method that leverages both historical data and real-time feedback. GNNs represent complex network topologies, while DRL refines routing strategies based on the evolving network state, leading to reduced packet loss and delay.

Ding et al. [113] propose "GROM", a generalised routing optimisation model that uses GNNs for structural network understanding and DRL for refining routing strategies with real-time data. The method achieves substantial improvements in throughput, latency, and bandwidth efficiency across different topologies.

Similarly, Almasan et al. [114] integrate GNN-derived embeddings with DRL agents to develop routing strategies capable of dynamically balancing latency and throughput in real-time conditions.

In addition to GNNs, CNNs have also been used alongside DRL. Tang et al. [17] design a DRL-CNN-based framework for routing in WMNs, where CNNs extract real-time features and DRL learns efficient routing policies. This solution significantly improves delay and packet loss over protocols like OSPF and RIP.

Likewise, Swain et al. [115] propose "CoDRL" for SDNs, combining CNNs for spatial feature extraction with DRL for policy refinement. The approach adapts to varying traffic patterns, achieving better throughput and reduced latency.

Alternative combinations have also been explored. Altamirano et al. [116] integrate DRL with Generative Adversarial Networks (GANs), where GANs generate diverse training data to enhance the DRL agent's robustness in SDN routing.

In distributed wireless networks, You et al. [117] propose DQRC, a multi-agent DRL method using LSTM-based RNNs. This allows each router to learn routing policies independently, balancing congestion awareness and path optimality, and resulting in improved delivery performance in dynamic conditions.

Rao et al. [118] present a distinct approach by combining DL with Lagrange multipliers to solve constrained routing problems in peer-to-peer networks. Their model adapts to changing conditions while satisfying latency and cost constraints, demonstrating the flexibility of DL for multi-objective routing optimisation.

3.3.6 Research Gap

The reviewed literature demonstrates the significant potential of ML and DL techniques to enhance network routing, particularly in terms of reliability, energy efficiency, and adaptability to dynamic conditions. However, most existing works employ learning models merely as auxiliary components within traditional routing architectures. These models are typically tasked with refining specific operations, such as selecting optimal paths, estimating trust, or mitigating congestion, while the core routing logic remains rule-based and protocol-driven.

This reliance on conventional protocols limits the flexibility and scalability of ML-based solutions, especially in dynamic or heterogeneous environments. Existing approaches often focus on optimising isolated metrics, such as energy or delay, rather than rethinking routing as a unified, predictive process. Few studies explore routing as a purely model-driven task that discards fixed rules in favour of real-time, local decision-making.

This thesis addresses this unexplored area by proposing a novel routing paradigm that replaces predefined protocol logic with a data-driven, predictive approach. It aims to treat packet forwarding as an intelligent decision process guided by real-time network context, offering improved adaptability and reduced overhead in dynamic IoT environments.

3.4 Conclusion

This chapter has reviewed a wide spectrum of ML and DL techniques applied to routing across diverse network environments. From early RL strategies and classical supervised models to advanced DL architectures, these approaches address key challenges such as link instability, energy efficiency, latency, and scalability.

While the surveyed work reflects a growing shift toward data-driven, adaptive routing, most efforts remain tied to enhancing existing protocol structures. This limitation underscores the need to rethink routing as a purely model-driven process, an approach explored in the following chapters.

Chapter 4

Machine learning based routing protocol (MLBRP) for Mobile IoT networks

4.1 Introduction

This chapter presents the first contribution of this thesis, which involves the development of a ML-based routing strategy for IoT networks. The proposed approach replaces conventional protocol-driven routing decisions with a predictive model that selects the next hop based on local observations and previous routing experiences.

Section 1 introduces the overall structure of the intelligent routing framework. Section 2 outlines the design, training, and validation process of the selected ML models. Section 3 describes the simulation environment and presents the performance evaluation results. Finally, Section 4 concludes the chapter and suggests potential directions for future research.

4.2 Problem definition and design objectives

Efficient routing in the IoT remains a significant challenge due to the dynamic and resource-constrained nature of IoT environments. Devices in these networks often operate with limited energy, processing power, and memory, and are deployed in scenarios characterised by frequent topology changes and unpredictable connectivity. Traditional routing protocols, such as AODV, rely heavily on broadcasting control packets for path discovery and maintenance, without considering the context of the routing situation or learning from previous interac-

tions. This leads to repetitive operations, redundant message exchanges, increased overhead, and wasted energy, especially in networks where similar routing scenarios occur frequently. These protocols treat each route discovery as an isolated event, missing the opportunity to optimise routing through knowledge gained from past experiences. Additionally, existing solutions lack the capability to adapt to the temporal and spatial regularities present in smart city environments, where mobility patterns often follow predictable routines. The absence of intelligence in conventional routing mechanisms results in poor scalability and inefficient use of resources, motivating the need for a more adaptive, context-aware approach. This work addresses these issues by proposing a ML-based routing solution capable of learning from previous routing decisions and local environmental observations, allowing each node to make informed, efficient forwarding decisions without relying on repeated network-wide control exchanges.

The limitations identified above form the foundation for the proposed MLBRP protocol. The primary design objectives of MLBRP are:

- **Learn from past experiences**: Capture historical routing data that reflects successful forwarding decisions, network topology changes, and contextual variables (e.g., time and position).
- **Exploit temporal and spatial patterns**: Use periodic observations (such as time of day and node position) to improve next-hop prediction accuracy based on regular mobility trends.
- **Reduce broadcast-based overhead**: Replace the repetitive and energy-consuming control message exchange with local, model-driven predictions.
- **Enable autonomous decisions**: Allow each node to make routing decisions independently by querying a trained ML model instead of communicating with neighbouring nodes.
- **Ensure robustness via fallback mechanisms**: Retain a traditional protocol as a safety net in the event of model uncertainty or failure.
- **Adapt to evolving network dynamics**: Support retraining or refinement of models to accommodate new patterns or topology changes over time.

By meeting these objectives, MLBRP offers a learning-oriented and context-sensitive alternative to conventional routing approaches. It provides a promising solution to extend

the operational lifespan of IoT devices, enhance network scalability, and support the real-time demands of smart environments.

4.3 Proposed ML-Based Routing Protocol (MLBRP)

The first contribution proposes a novel framework developed to improve data packet routing in IoT networks by leveraging the predictive power of ML models trained on historical routing decisions. By enabling network nodes to make autonomous, data-driven routing choices, the framework minimises the frequency of communication with neighbouring nodes during route discovery and maintenance. This not only reduces protocol overhead but also improves routing efficiency, particularly in dynamic and resource-constrained environments, such as those found in smart city deployments.

At the core of this framework is the idea that nodes can learn from past routing experiences. Each participating node maintains a localised history of its successful packet-forwarding decisions, capturing relevant contextual data with each instance. This accumulated dataset is then analysed to identify routing patterns and train predictive models that assist in next-hop selection. Unlike traditional routing protocols, which rely heavily on real-time communication with neighbouring nodes, our approach allows nodes to consult their trained models for predictive routing, significantly reducing dependency on immediate neighbour queries.

Importantly, while the ultimate goal is to transition fully to model-driven routing, traditional routing protocols play a crucial role during the system's initialisation. They are responsible for generating the foundational data required for model training. Once a sufficient volume of meaningful routing data is collected and processed, these protocols can be gradually phased out in favour of autonomous, machine-learning-based decision-making.

The proposed framework is structured into three main phases, each corresponding to a key stage in the transformation from conventional routing to intelligent, model-based routing. These phases are detailed in the following sections and illustrated in Figure 4.1.

4.3.1 System overview

The operation of the proposed system is structured into three sequential phases, enabling a progressive shift from traditional routing protocols to a ML-based routing mechanism. In the early phase, conventional protocols are used not only to manage packet forwarding but also to collect valuable routing data from the network environment. This data, consisting of

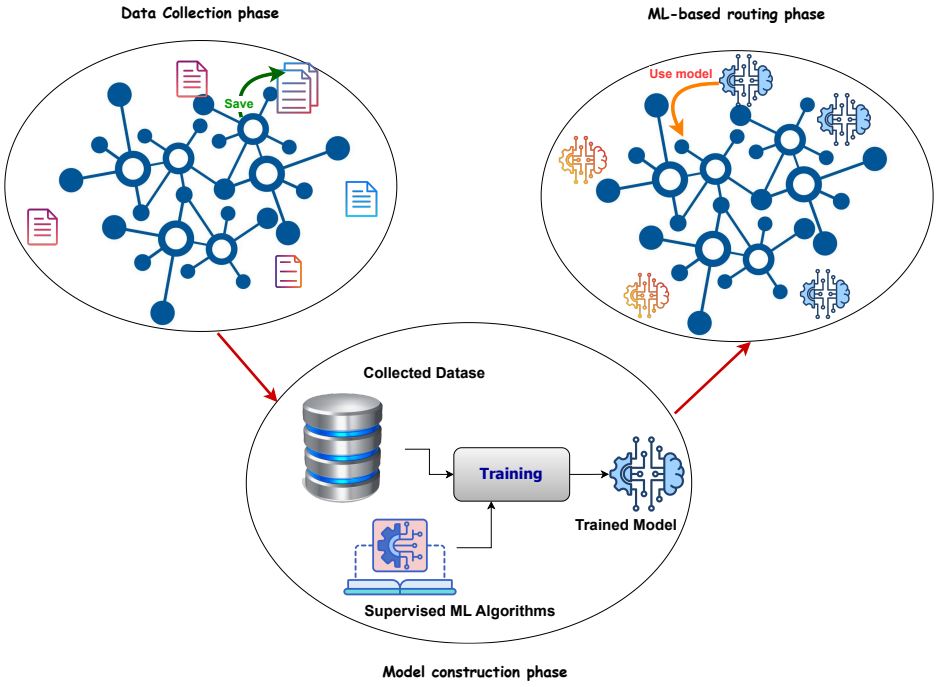

Figure 4.1: System model

successful routing decisions, is accumulated and analysed to train the predictive model. As the model becomes more reliable, it gradually assumes responsibility for routing decisions, reducing the system's reliance on real-time communication with neighbouring nodes. The three phases, each addressing a specific stage in this transition, are described in detail in the following subsections.

4.3.1.1 Data collection phase

A central element of our approach is the systematic construction of a dataset that precisely reflects the routing decision-making behaviour of network nodes during packet forwarding. In contrast to externally sourced datasets, this dataset is dynamically generated as an inherent part of the network's functioning. Each node continuously logs its successful routing decisions, along with the contextual factors that influenced those decisions, and stores this information in a dedicated local database. The resulting dataset forms the foundation for training intelligent models that guide future routing decisions.

A. Dynamic nature of dataset generation

Due to the variability in network topologies, mobility patterns, and traffic dynamics, the

datasets generated across different networks naturally exhibit significant differences. This diversity highlights the necessity of a routing model capable of adapting to the specific conditions of each network environment. For example, a dataset obtained from a densely populated urban IoT scenario, characterised by frequent node mobility, will reflect routing behaviours distinct from those observed in a static sensor network with a fixed topology.

As such, dataset generation cannot follow a uniform approach; it must be inherently aligned with the dynamic characteristics of the network in question. To achieve effective adaptability, our method involves collecting training data directly from the target network during an initial setup phase. This strategy enables the routing model to learn and adjust to the network's specific features without depending on externally prepared datasets.

B. Data collection process

The data collection process is initiated by operating the network under standard conditions, during which a conventional routing protocol is responsible for managing packet forwarding. At this stage, the traditional protocol plays a critical role, as its routing decisions form the basis of the dataset. Whenever a packet is successfully transmitted, the node records the selected next hop along with the contextual factors that influenced the decision. Through continuous logging over time, a comprehensive dataset is constructed, capturing the routing behaviour as it naturally occurs within the network. Once a sufficient volume of data has been gathered, this dataset serves as the foundation for training a ML model capable of accurately predicting optimal next-hop selections.

C. Features recorded in the dataset

In typical network operations, routing decisions are guided by a protocol that determines the most suitable path for forwarding packets. At each node, this protocol identifies the next hop, facilitating the packet's journey from source to destination through multiple intermediate nodes. However, in mobile environments, frequent changes in node positions alter network connectivity over time, making routing decisions highly sensitive to temporal, spatial, and local network conditions.

Despite this dynamic behaviour, many nodes exhibit regular mobility patterns, introducing a degree of predictability into the routing process. For instance, a student who regularly attends a class every Saturday at 10 AM or a doctor who routinely contacts a patient on Monday mornings may repeatedly interact with the same neighbouring nodes at specific times. Similarly, vehicles following scheduled routes and individuals commuting at fixed hours contribute to recurring communication patterns within the network. These periodic behaviours

present an opportunity to apply ML techniques to identify and exploit such patterns for more efficient routing.

To develop a dataset that accurately reflects these characteristics, each node logs a set of relevant features following every successful routing decision:

- **Actual node**: The node currently in possession of the data packet and making the routing decision.
- **Previous node:** The node from which the packet was received.
- **Source node:** The originator of the packet transmission.
- **Destination node:** The final intended recipient of the packet.
- **Actual neighbours :** The set of nodes within the transmission range of the current node at the time of decision-making.
- **Time:** The precise timestamp of the routing decision, including the day of the week, calendar date, and time of day. This feature is critical for capturing periodic patterns in node interactions and for analysing temporal variations in routing decisions. For example, in a smart city context, nodes may exhibit more predictable behaviour during peak traffic hours than during off-peak times.
- **position:** The geographic coordinates of the node at the moment of decision-making. Since node mobility influences routing, tracking location over time helps identify spatial trends and enhance prediction accuracy.
- **Next hop:** The selected neighbouring node to which the packet is forwarded. This value serves as the target variable during the training phase, rather than an input feature.

These features are deliberately selected to capture both static attributes and dynamic, time-varying influences on routing decisions. In particular, the inclusion of temporal and spatial data allows the ML model to exploit patterns that emerge from regular node behaviours, such as vehicles on fixed routes, employees commuting at fixed times, or smart devices performing routine tasks, thereby enhancing prediction accuracy.

D. Outcome of the data collection phase

This phase results in the creation of multiple datasets, each systematically compiled after collecting a sufficient volume of routing-related data from participating nodes. These datasets form the essential groundwork for developing intelligent routing models in the subsequent phases. By ensuring that the data faithfully reflects the actual operational behaviour of the network, we facilitate the construction of models capable of making adaptive, context-aware routing decisions. This, in turn, contributes to enhanced overall network performance and routing efficiency.

4.3.1.2 Model construction phase

Following data collection and preprocessing, the next step involves constructing decision models tailored to each network node. The central aim of this phase is to develop a dedicated classifier for every participating node, enabling it to accurately determine the most appropriate neighbouring node to serve as the next hop for data packet forwarding.

To accomplish this, we employ supervised ML techniques trained on historical routing data collected from the nodes. The selected algorithms include DTs, known for their interpretability through rule-based structures; artificial neural networks, capable of modelling intricate patterns and nonlinear relationships; and SVMs, which perform well in high-dimensional feature spaces. Each algorithm offers distinct advantages in terms of predictive accuracy, transparency, and computational demand.

These models are trained to recognise the decision-making patterns embedded in the collected dataset. Once deployed, each node utilises its trained model to autonomously assess its current context and select the optimal next hop. This model-driven routing approach enhances both the efficiency and adaptability of packet forwarding across the network.

4.3.1.3 ML-based routing phase

At this stage, each network node transitions from relying on conventional routing protocols to employing its trained ML model for making routing decisions. The ML model becomes the principal mechanism for routing decisions, utilising inputs such as temporal indicators, node-specific attributes, and packet-related features to determine the most suitable next hop for forwarding data packets. These decisions are guided by the patterns and relationships learned during the training phase.

By leveraging the ML model, nodes are able to make routing decisions that are both context-sensitive and data-driven, thereby reducing dependence on traditional, protocol-based mechanisms. This shift is expected to enhance the network's efficiency and adapt-

ability, particularly in dynamic environments.

Nonetheless, the inherently unpredictable nature of mobile and distributed networks means that the model's predictions may occasionally result in suboptimal or unsuccessful routing outcomes. To address this, a **fallback mechanism** is incorporated: when the ML model fails to identify a viable next hop, the node reverts to the traditional routing protocol. This hybrid strategy ensures the robustness of the routing process, combining the adaptability of ML with the reliability of conventional methods.

4.3.2 Execution modes of the proposed framework

Our proposed solution is structured into three main phases: data collection, model training, and ML-based routing. To ensure adaptability and efficient use of available computational resources, these phases can be executed in one of two distinct modes: the **local processing mode** and the **distant processing mode**. The selection of the mode is dynamically determined based on the resource availability at the node level.

4.3.2.1 Local processing mode

In this mode, all three phases, including the training of the ML model, are carried out locally by the node itself. This is suitable for scenarios where the node has sufficient computational power, memory, and energy to perform the operations autonomously. It promotes decentralisation, reduces dependency on external systems, and can result in lower latency.

A. Data collection phase

In this phase, each node continuously logs essential information following every successful routing decision. The recorded data includes the node's current status, characteristics of the transmitted packet, and prevailing network conditions at the time of decision-making. As these entries accumulate over time, they form a rich dataset that encapsulates the node's operational context and routing behaviour. This dataset, illustrated in Figure 4.2, serves as the foundation for training the ML model in the next phase.

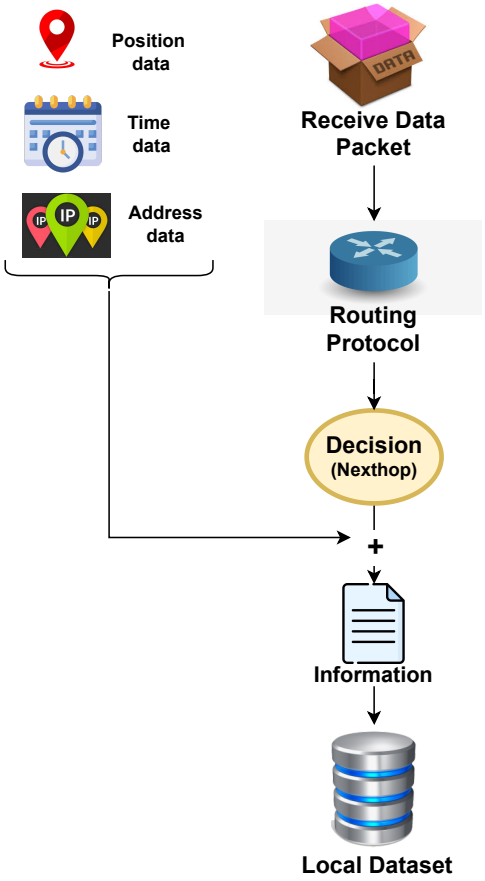

Figure 4.2: Data collection phase in local processing mode

B. Model construction phase

Following the accumulation of sufficient routing data, the node advances to the model construction phase. During this stage, the node applies a chosen ML algorithm to train and validate a predictive routing model. The training process is conducted using the dataset gathered earlier, enabling the model to extract patterns and infer relationships based on historical routing behaviors. Validation is then performed to assess the model's accuracy and its ability to generalize to new, unseen scenarios, thereby ensuring its dependability in making future routing decisions (refer to Figure 4.3).

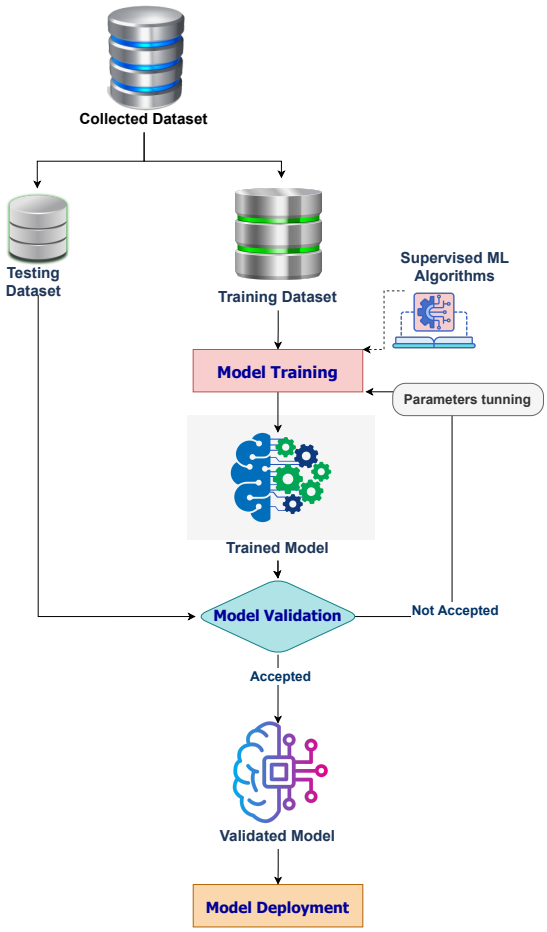

Figure 4.3: Model construction phase in local processing mode

C. ML-based routing phase

Once the decision model has been properly trained and validated, the node begins using it to make real-time routing decisions (see Figure 4.4). In this phase, the node replaces traditional routing protocols with the trained ML model to select the most appropriate next hop for each data packet. By leveraging learned patterns and contextual insights, the model enables more adaptive and intelligent routing, improving overall efficiency and responsiveness to changing network conditions.

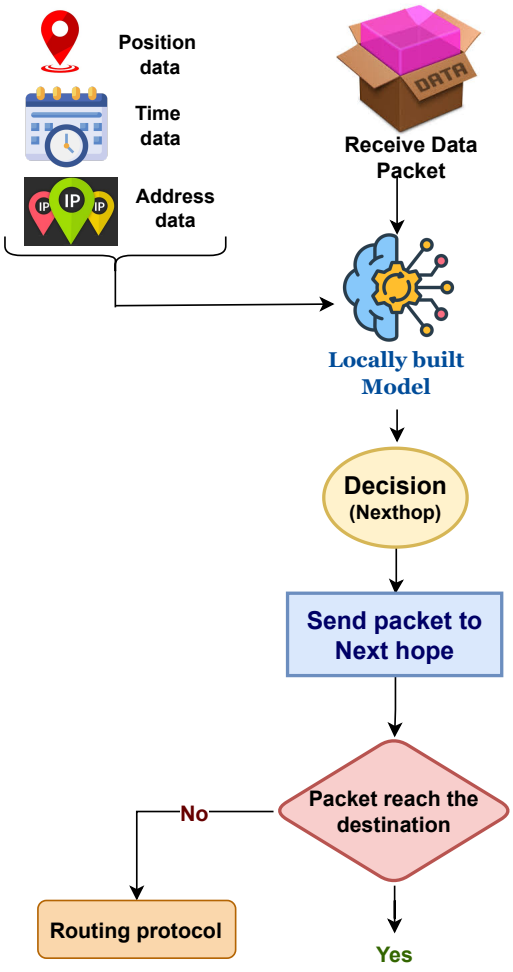

Figure 4.4: ML-based routing phase in local processing mode

4.3.2.2 Distant processing mode

When a node lacks the necessary resources to perform local model training, it switches to the distant processing mode. In this mode, the node collects routing-related data and sends it to a more powerful external computing entity, such as a cloud server or an edge node. These servers aggregate datasets from multiple nodes across the network, handle preprocessing and other resource-intensive operations, and then construct and validate ML models tailored for each node. Once training is complete, the resulting models are transmitted back to the originating nodes, where they are used to support intelligent routing decisions. This approach

allows even resource-constrained nodes to benefit from ML-based optimisation without the overhead of local computation.

Figure 4.5 illustrates the proposed distant processing mode.

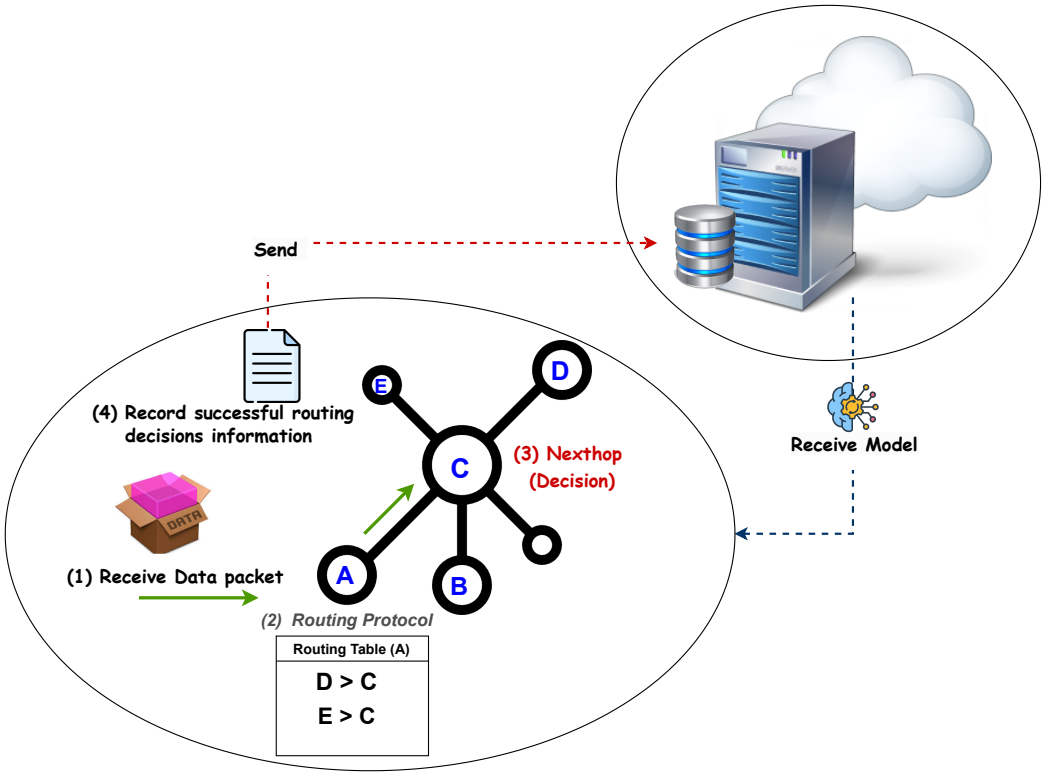

Figure 4.5: Distant processing mode

A. Data collection phase

In this mode, every time a node completes a successful routing decision, it promptly transmits the associated information to a remote server. This centralised server, designed with superior computational and storage capabilities, gathers data from multiple nodes across the network. It is responsible not only for aggregating this data but also for preprocessing it, ensuring the resulting datasets are clean and well-structured for the subsequent model training and validation stages (see Figure 4.6).

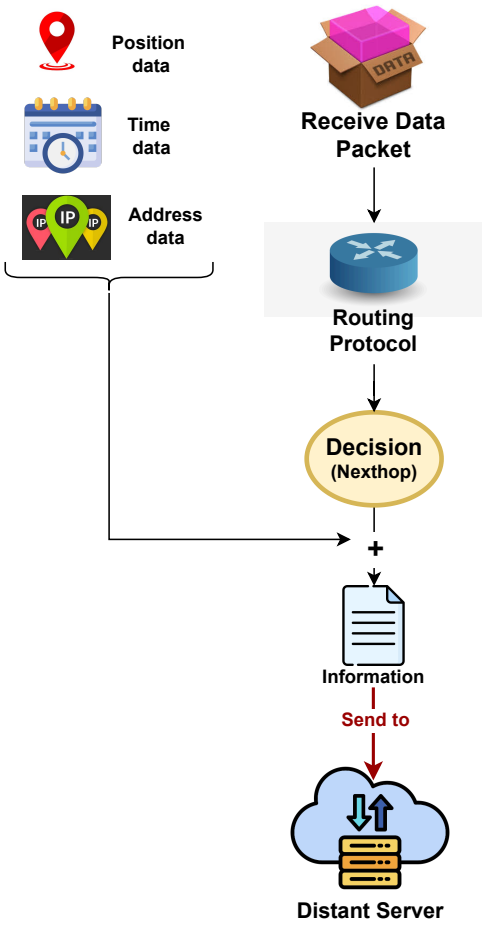

Figure 4.6: Data collection phase in distant processing mode

B. Model construction phase

After collecting and aggregating datasets from the nodes, the remote server initiates the ML phase. It constructs and validates a tailored decision model for each node based on its unique routing data and contextual network conditions. This process includes choosing suitable ML algorithms, training the models on the gathered datasets, and validating their performance to ensure they can generalise well to new scenarios and make dependable routing decisions.

Once the models have been successfully trained and validated, the server transmits them back to the corresponding nodes. These models then replace traditional routing mechanisms

at each node, enabling more intelligent, adaptive, and efficient packet forwarding across the network (see Figure 4.7).

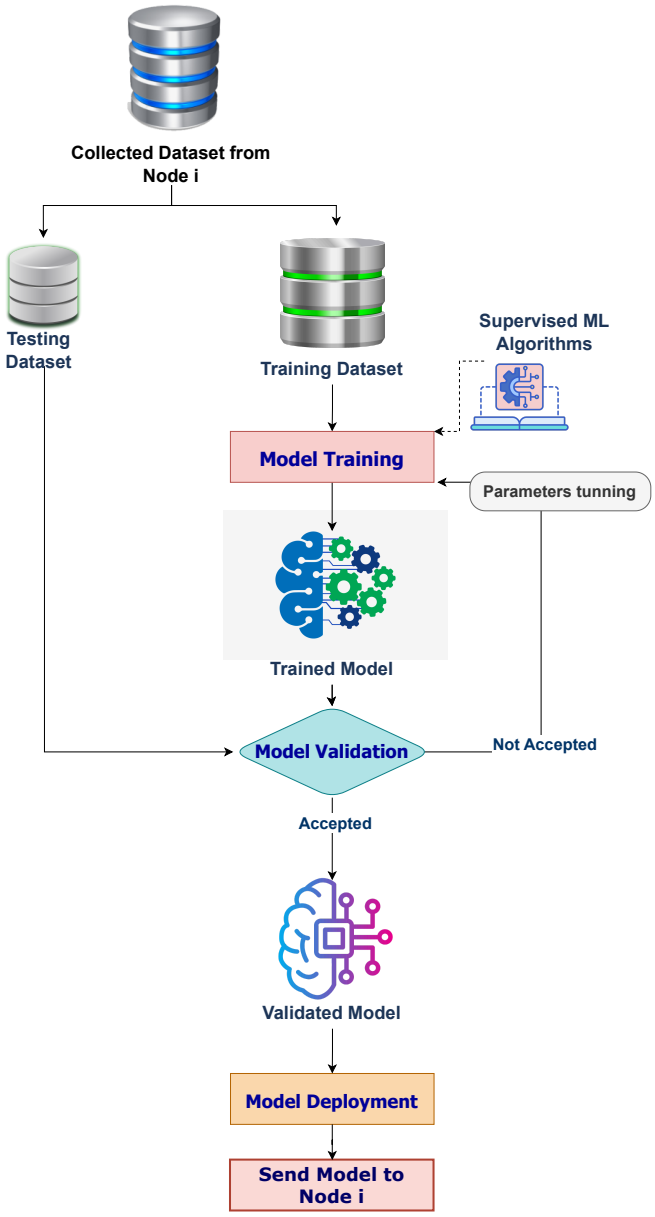

Figure 4.7: Model construction phase in distant processing mode

C. ML-based routing phase

After receiving their corresponding ML models from the server, the nodes start applying

these models to select the most suitable next hop for each packet. The decision-making process is now driven by learned insights rather than predefined routing rules, allowing the nodes to adapt to dynamic network conditions and operate more intelligently.

To ensure reliability, a fallback mechanism is incorporated. If the ML-based decision-making fails, due to model inaccuracies, abrupt topology changes, or other unforeseen issues, the node automatically reverts to the original traditional routing protocol. This hybrid strategy balances innovation with resilience, ensuring continuous and stable network performance even when the ML approach encounters limitations (see Figure 4.8).

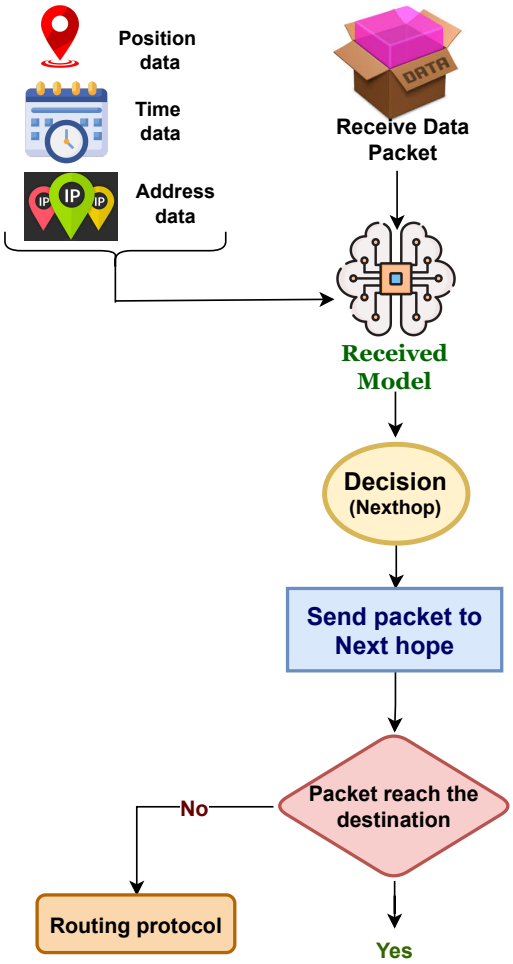

Figure 4.8: ML-based routing phase in distant processing mode

This dual-mode execution capability enhances the scalability and practicality of our routing framework, making it suitable for heterogeneous network environments where nodes may have varying capabilities.

4.4 Simulation framework and experimental methodology

In the absence of a real-world IoT testbed equipped with the necessary infrastructure and node availability, this study adopts a simulation-driven approach to validate the proposed ML-based routing strategy. Simulations offer a flexible and controlled environment to emulate the complex and dynamic nature of mobile IoT networks, thereby enabling the generation and collection of meaningful data to support both training and evaluation processes.

To carry out this investigation, we employed NS2, a widely recognised and extensively utilised tool in the networking research community. NS2 provides a comprehensive framework for simulating protocol behaviour, node mobility, and communication dynamics, making it a suitable choice for this study.

The evaluation methodology is structured into three fundamental phases: data acquisition, ML model development, and performance assessment. These stages are outlined in detail below.

Phase I: Data collection

- In this phase, simulation scenarios were created using NS2 to model the movement and communications of mobile nodes over a specific period.
- The AODV routing protocol was implemented as the baseline mechanism to handle routing operations throughout the simulated duration.
- During the simulation, routing decisions and contextual parameters, including timestamps, node identifiers, and the local network environment, were logged. This dataset served as the foundation for training the ML models.

Phase II: Model construction

- The collected dataset was then used to train a set of supervised learning algorithms, namely: **SVM**, **DT**, and **ANN**.

- These models were tailored to predict the optimal next-hop node by learning from the decision-making patterns observed in the original AODV protocol.
- The goal of this phase was to enable the ML models to function as an intelligent routing approach capable of dynamically selecting efficient paths within the network.

Phase III: Performance evaluation

- To assess the effectiveness of the ML-based routing models, we conducted a comparative analysis between two routing scenarios: the traditional AODV protocol and the ML-based decision-making approach.
- The evaluation metric used was the **total number of control messages** exchanged during the routing process, as observed in the dataset.
- The degree of reduction in control message overhead served as an indicator of the performance improvements introduced by the ML-based routing strategy.

The subsequent sections provide a comprehensive explanation of the simulation parameters, the configuration of the baseline routing protocol, and the performance indicators used to benchmark and compare the routing strategies under investigation.

4.4.1 Simulation environment

To evaluate the effectiveness of the proposed ML-based routing mechanism for mobile IoT environments, we conducted a series of simulations using NS2 [119]. NS2 is a well-established, discrete-event simulation platform extensively used in the field of computer networking research. It provides a comprehensive framework for modelling various network protocols, including those for transport (such as TCP), routing, and multicast communication, applicable to both wired and wireless infrastructures.

The NS2 framework is composed of two core components: the Network Simulator (NS) and the Network Animator (NAM). The NS component serves as the simulation engine, executing network protocol logic and handling the detailed events related to packet transmission, node mobility, and protocol behaviour. In contrast, NAM functions as a visualisation tool, offering an animated graphical interface that helps researchers observe the dynamics of network behaviour and verify simulation outcomes.

The simulator is built using a hybrid architecture comprising C++ for performance-intensive backend functionalities and Object-oriented Tool Command Language (OTcl) for

scripting and scenario configuration. This separation allows for efficient simulations while offering flexibility in modifying and controlling simulation scenarios. Final outputs from the simulation are rendered and visualised through NAM, enabling clear analysis of node interactions and protocol operations throughout the simulated experiments.

4.4.2 Reference routing protocol for decision logging

Although any particular routing protocol does not constrain our proposed approach, we employed the AODV protocol as a reference model for routing behaviour. AODV was selected due to its simplicity, widespread adoption, and availability of well-documented implementations [76]. In our methodology, AODV served as the decision-making engine whose routing actions were observed and logged to construct the dataset used for training our ML models.

AODV is a reactive routing protocol designed for dynamic, infrastructure-less networks such as MANETs and mobile IoT systems, which may consist of tens to thousands of mobile nodes. Unlike proactive protocols, AODV initiates route discovery only when data needs to be transmitted. When a source node has data to send, it first checks its routing table for a valid path to the destination. If none exists, a route discovery process is initiated using two types of control messages: Route Requests (RREQs) and Route Replies (RREPs).

In this process, the source node broadcasts an RREQ to its immediate neighbours. If a neighbour does not possess a valid route or is not the destination, it records a reverse path entry and rebroadcasts the RREQ further into the network (see Figure 4.9).

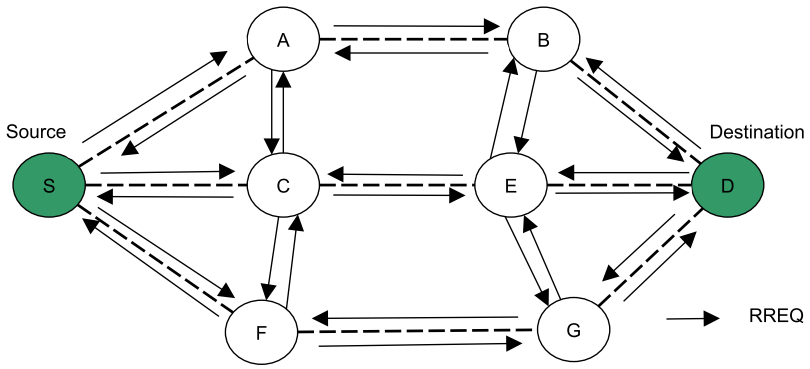

Figure 4.9: Dissemination of the RREQ messages

Once the RREQ reaches the destination or an intermediate node with a valid route, an RREP is generated and sent back to the source along the reverse path (see Figure 4.10). After receiving the RREP, the source node finalises the route and begins forwarding data

packets.

To maintain connectivity and adapt to dynamic network topologies, AODV also uses periodic Hello messages, typically sent every second, to monitor the availability of neighbouring links.

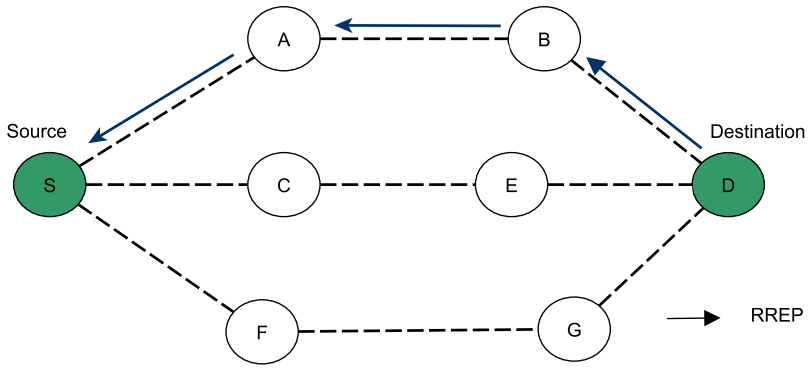

Figure 4.10: Dissemination of the RREP messages

A. Cost estimation of transmitting a single data packet using the AODV protocol

To estimate the communication overhead and operational efficiency of the AODV routing protocol, we conduct an analytical evaluation of the total number of **control and data messages** exchanged during the route discovery and data transmission process for a single packet.

This estimation serves as a reference point to assess the cost, effectiveness of the AODV protocol. More importantly, it provides a basis for comparing AODV's communication overhead with that of our proposed model, based routing approach, which bypasses the traditional route discovery phase and does not rely on control messages such as RREQ and RREP.

The analysis is based on the following simplified assumptions:

- The network consists of n nodes.
- Each node has an average of k neighbours, with k assumed to be 4.
- The maximum number of hops from a source to a destination node, m, is approximated as $\sqrt{n}$.

We break down the analysis as follows:

a) **Total number of transmitted RREQ messages:** The route discovery process begins with the source node broadcasting a single RREQ message to all its k neighbouring nodes. Each recipient node rebroadcasts the RREQ to its neighbours, and this process continues until the message has been disseminated throughout the network. Assuming full reachability and no redundant suppression, this results in approximately n RREQ transmissions across the network.

b) **Total number of transmitted RREP messages:** When the RREQ reaches the destination node or an intermediate node with a valid route, an RREP message is generated and unicasted back to the source along the reverse path. Given that the average path length is m hops, this results in m RREP transmissions.

c) **Total number of transmitted data packets:** After the route is established, the source node begins data transmission along the discovered path. Since the path consists of m hops, forwarding a single data packet to the destination involves m transmissions.

The overall count of transmitted messages (T), which includes both control messages for route discovery and the actual data packet transmission, can be defined as:

$$T = n_{\text{RREQ}} + m_{\text{RREP}} + m_{\text{DATA}} \tag{4.1}$$

In parallel, the total number of received messages (disregarding message size) is calculated by:

$$k \times n + 2 \times m \tag{4.2}$$

Where:
$k \times n$: Total number of received RREQ messages.
m: Total number of received RREP messages or data packets.

4.4.3 Simulation context and mobility model

A. Simulation context: The simulations were conducted using the NS-2 network simulator. A total of 31 nodes were deployed within a rectangular area of 3500 m × 900 m. Each simulation scenario was executed over 150 seconds per day. The network topology used in the simulations is shown in Figure 4.11. TCP traffic was employed as the transport protocol, and all data packets were fixed at 512 bytes in size. Two node

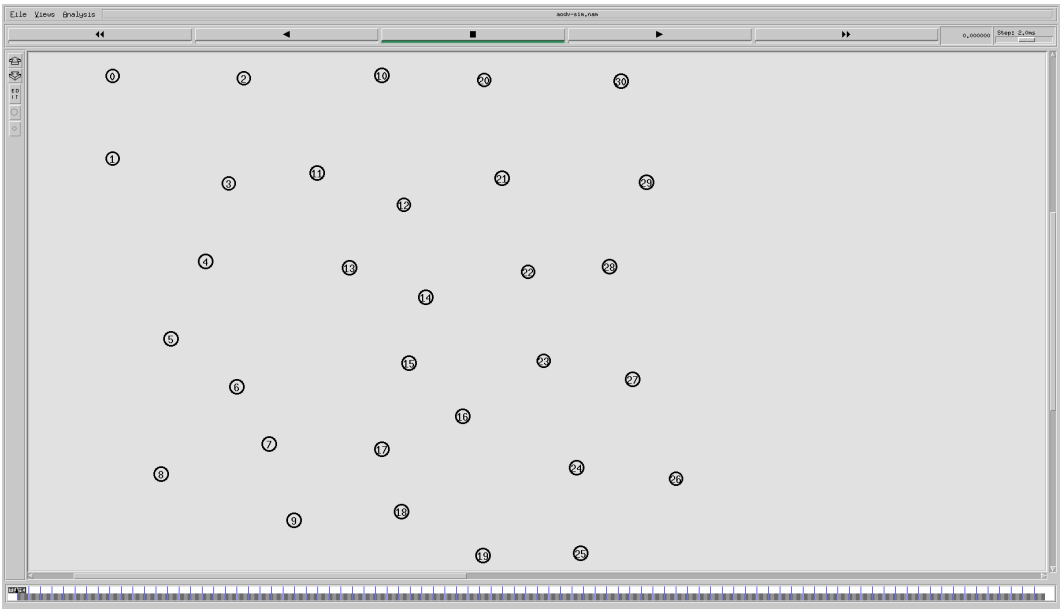

Figure 4.11: The created network topology

speed settings, 10 m/s and 15 m/s, were used to evaluate mobility effects. Table 4.1 summarises the simulation parameters used in the study.

B. Mobility model: The mobility pattern implemented in the simulations combines both regular and random movement components to emulate realistic IoT scenarios. Regular movement ensures that nodes periodically come into each other's transmission range, simulating scheduled or routine paths typical in smart city settings (e.g., public transportation routes or pedestrian flows). In contrast, the random movement component introduces occasional unpredictability, capturing the irregular and spontaneous behaviour commonly seen in mobile environments. This hybrid mobility model enables a comprehensive evaluation of the routing protocol under both structured and dynamic movement conditions.

To enable a fair and meaningful comparison between our ML-based routing approach and the AODV protocol, we carefully selected simulation parameters that strike a balance between representativeness and computational tractability. A moderate-sized network (comprising 31 nodes) was employed to capture essential dynamics without incurring excessive complexity. Other parameters, such as node speed, simulation time, and topology dimensions, were chosen within realistic operational bounds. Since both routing methods were evaluated under the same conditions, the absolute parameter values do not affect the comparative results. Instead,

Table 4.1: Simulation configuration parameters

Parameter	**Value**
Channel type	Wireless
Propagation model	Two-Ray Ground
MAC type	IEEE 802.11
Link layer type	LL
Interface queue type	DropTail/PriQueue
Number of nodes	31
Simulation duration	150 seconds
Routing protocol	AODV
Topology dimensions (X, Y)	3500 m, 900 m
Mobility model	Daily regularly repeated
Traffic type	TCP
Node speeds	10–15 m/s

the focus is on measuring the relative performance improvement achieved by the proposed model-based routing technique over the traditional AODV protocol.

4.4.4 Collected dataset

Following the establishment of the network and the integration of the defined mobility model, simulations were executed over a continuous 7-day period. During this time, we systematically recorded successful routing decisions made across the network. These decisions serve as critical indicators for assessing the performance of the routing protocol under diverse operational conditions.

The recorded data were structured into five key features: Day, Current Time, Current Node (Node-ID), Destination Node, and Next-Hop Class. The "Next-Hop Class" denotes the selected forwarding node for a given routing decision, with the number of possible classes corresponding to the total number of nodes in the network. Before use, the dataset was cleaned by removing duplicate and incomplete entries, ensuring consistency and reliability.

The dataset was stored in **ARFF** (Attribute-Relation File Format), which is compatible with the **WEKA** platform. Although most values appear numeric, they **represent categorical** identifiers, except for the 'Current Time' attribute, which is **continuous**.

In total, the dataset comprises 16,384 labelled instances, aggregated from all nodes, and

serves as the foundation for training and evaluating the ML models in subsequent stages.

4.5 Model construction

To construct and evaluate the predictive capabilities of the proposed ML models, we employed the Weka software toolkit [120], a widely recognised open-source platform that provides an extensive collection of ML algorithms and data preprocessing utilities tailored for data mining tasks. Weka's built-in classification and validation functions were utilised to train our models and evaluate their predictive performance, particularly in terms of accuracy and generalisation.

Weka was chosen for its ease of use, visual interface, and flexibility in experimenting with different algorithms and validation techniques, making it particularly suitable for rapid prototyping and academic research in ML.

4.5.1 Tested machine learning methods

Among the range of supervised learning algorithms available, we selected DT, SVM, and ANN based on their proven effectiveness in balancing interpretability, computational efficiency, and predictive accuracy, all of which are critical factors in the context of resource-constrained IoT environments.

We selected these models based on the following criteria:

- **Generalisation ability:** Refers to the model's capacity to identify meaningful patterns in the training data and apply them effectively to new, unseen instances, an essential trait for ensuring reliable routing decisions in dynamic environments.
- **Training efficiency:** Denotes the time and computational resources required to build the model from the training dataset. Fast training is particularly beneficial during frequent model updates or when operating within constrained time windows.
- **Inference speed:** Represents the time needed to classify a new input instance. This is a critical factor in real-time routing scenarios, where delayed decisions can negatively impact network responsiveness and performance.
- **Model complexity and size:** Involves the structural intricacy and memory requirements of the model. A lightweight model is preferable for deployment on IoT devices with limited processing power and storage capacity.

Each algorithm presents inherent trade-offs across these dimensions. To optimise performance, we performed **hyperparameter tuning** for each model: adjusting tree depth in DTs, selecting appropriate kernels for SVMs, and configuring the architecture (number of layers and neurons) for ANNs.

Although the models vary in complexity and generalisation behaviour depending on the nature of the input data, the primary performance metric for this study was routing accuracy, specifically, the precision in predicting the correct next-hop node. Accurate next-hop selection is essential for enhancing routing efficiency and reducing overhead.

Through this comparative evaluation, we demonstrate the feasibility and effectiveness of incorporating ML-based decision-making into routing protocols, presenting it as a compelling alternative to traditional approaches like AODV.

4.5.2 Validation and evaluation metrics

This section outlines the validation strategy adopted to assess the performance of the proposed ML models, along with the evaluation metrics used to compare our approach with the baseline AODV protocol.

4.5.2.1 N-fold cross-validation

To evaluate the generalisation capability of the proposed models, we employed N-fold cross-validation [59], with $N = 10$. In this method, the dataset is partitioned into k disjoint subsets $D_1, D_2, \ldots, D_k$ of equal size. The model is trained and tested k times, each time using a different subset D_i for testing and the remaining $k - 1$ subsets for training. This iterative process ensures that every instance in the dataset is used for both training and evaluation. The final model accuracy is computed as the average of the accuracies obtained in each iteration. Table 4.2 illustrates this process.

Table 4.2: N-cross validation

	D1	**D2**	**...**	**Dn**
Iter 1	**Test**	Training	...	Training
Iter 2	Training	**Test**	...	Training
...	Training	Training	**Test**	...
Iter n	Training	Training	...	**Test**

4.5.2.2 Evaluation metrics

To evaluate and compare the performance of the proposed ML-based routing method (MLBR) against the AODV protocol, we define the following metrics:

A. Accuracy

Accuracy (P) denotes the proportion of correctly classified instances relative to the total number of instances evaluated. In the context of our work, it indicates the percentage of packets that are successfully routed compared to the total number of packets transmitted. The formula is given by:

$$P = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{4.3}$$

where

$$L = \begin{cases} 1 & \text{if } y_i = f(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

In this expression, x_i refers to the i-th packet, y_i represents the next hop determined by the AODV protocol for that packet, and $f(x_i)$ denotes the next hop predicted by our proposed model.

B. Sent packets (Sp)

The total number of transmitted packets is computed individually for both the proposed approach and the AODV protocol as follows:

a) **Proposed method (MLBR)**

The total number of sent packets (Sp_{MLBR}) consists of:

- Successfully transmitted packets,
- Failed transmissions requiring retransmission,
- Retransmissions handled by AODV.

The formula is given by:

$$sp_{MLBR}\begin{cases} = [succes] + [failed] + [retansmission] \\ = [p \times N \times m_{DATA}] + [(1-p) \times N \times m_{DATA}] \\ +[(1-p) \times N \times (n_{RREQ} + m_{RREP} + m_{DATA})] \\ = N \times m_{DATA} + (1-p) \times N \times (n_{RREQ} + m_{RREP} + m_{DATA}) \end{cases} \tag{4.5}$$

After simplification, it becomes:

$$Sp_{MLBR} = N \times m + (1-P) \times N \times (n + 2 \times m) \tag{4.6}$$

b) **AODV protocol**

Each data packet requires route discovery and reply operations in addition to transmission:

$$sp_{AODV} = N \times (n_{RREQ} + m_{RREP} + m_{DATA}) \tag{4.7}$$

After simplification, it becomes:

$$sp_{AODV} = N \times (n + 2 \times m) \tag{4.8}$$

C. Gained packets

The improvement in transmission efficiency, measured as the reduction in sent packets, is defined by:

$$Gain\begin{cases} = sp_{AODV} - sp_{MLBR} \\ = N \times (P \times n + 2 \times P \times m - m) \end{cases} \tag{4.9}$$

D. Received packets

This metric reflects the total number of packets successfully received during the transmission of N messages using our approach:

$$Received = N \times (m + (1-P) \times (n \times k + 2 \times m)) \tag{4.10}$$

Parameters

The following parameters are used in the above equations:

- N: Total number of data packets transmitted.
- P: Accuracy of the ML model.
- n: Number of nodes participating in route discovery.
- m: Average hop count per data transmission.
- k: Average number of routes evaluated during retransmission.

4.6 Results and discussion

Following the preparation of the training dataset, we employed the WEKA environment to develop our models using three distinct ML algorithms: DT, SVM, and ANN. Each model was trained on the selected routing-related features and subsequently evaluated to determine its effectiveness in predicting routing decisions within the IoT network. The performance outcomes are illustrated in Figure 4.12.

It is worth noting that the resulting model functions as a global training model, reflecting the aggregated performance of individual node models. This design ensures that the learned decision-making patterns are not confined to a single node but are instead applicable throughout the network, contributing to a more resilient and generalizable routing solution.

4.6.1 Analysis of models' performance

Among the evaluated models, the DT algorithm demonstrated the highest performance, achieving an accuracy of 87.78%. In comparison, the ANN and SVM models achieved accuracy rates of 71.6% and 83.1%, respectively.

An accuracy of 87.78% indicates that the MLBR protocol correctly predicted the next-hop node for 87.78% of the transmitted packets. The remaining 12.22% were misclassified, meaning those packets were routed to incorrect next hops and may not have immediately reached their destinations. Nevertheless, as outlined in our method, such misrouted packets are subsequently retransmitted using the AODV protocol, ensuring eventual delivery despite initial errors.

The outstanding performance of the DT model can be attributed to several advantages. Firstly, DTs are well-suited for handling symbolic and categorical data, such as current node

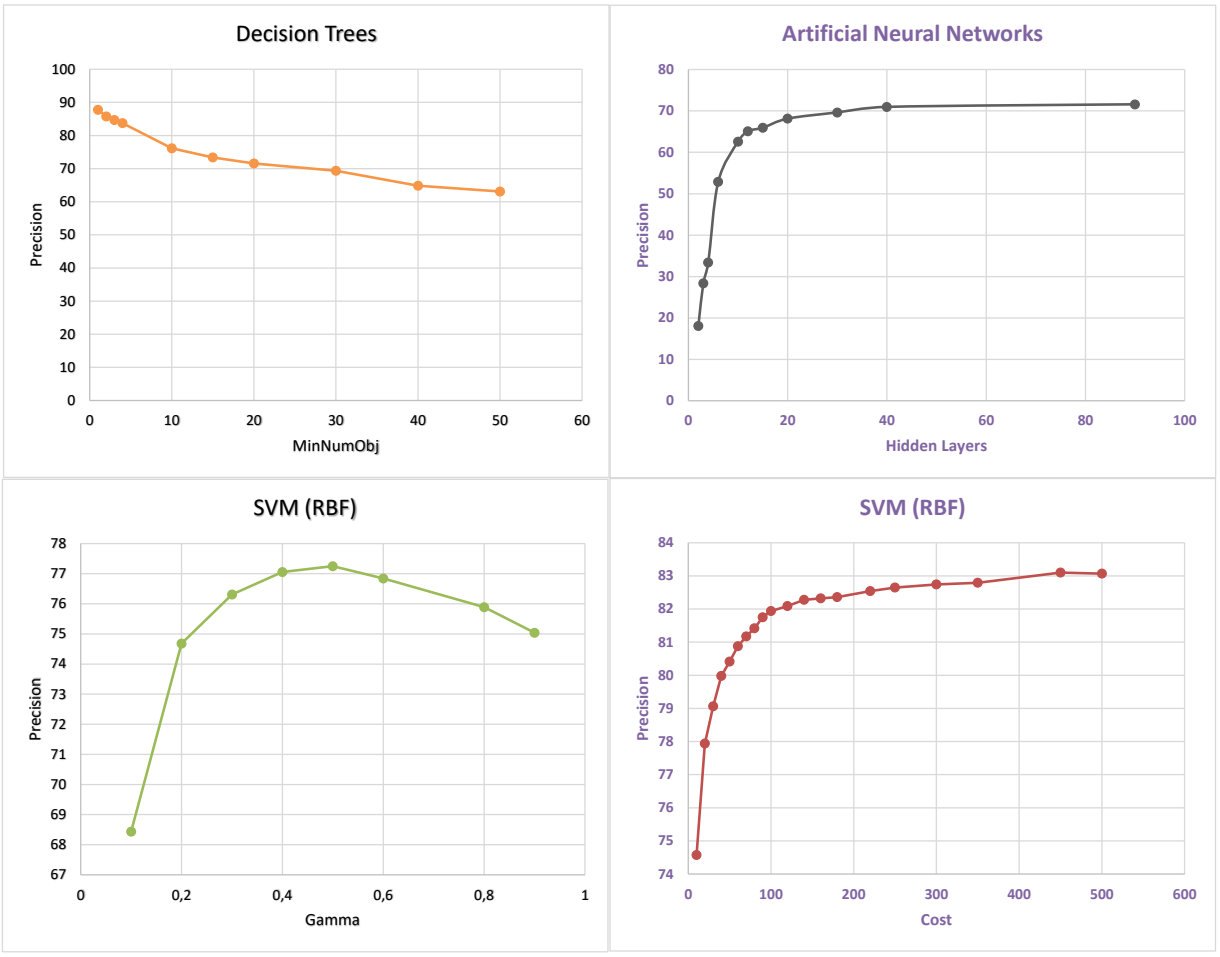

Figure 4.12: Accuracy obtained using different ML algorithms

ID, destination node, and next-hop, which are predominant in our dataset. Unlike models that require numerical inputs, DTs can directly process categorical variables without the need for encoding or normalisation, preserving the structure and semantics of the original routing data.

Secondly, the decision-making process in routing often follows hierarchical, rule-based logic, e.g., "If the current node is X and the destination is Y, then select next hop Z." This structure maps naturally onto the tree-like architecture of DTs, where branches represent conditions and outcomes. As a result, DTs can effectively model such relationships in a way that closely mirrors real-world routing logic. In contrast, models like ANN and SVM rely on continuous, often nonlinear decision boundaries, which may be less effective for learning and generalising rule-based, discrete decision patterns.

Ultimately, a model's effectiveness depends on two main aspects: (1) the nature of the dataset and (2) the structural and computational characteristics of the model. Given that

our dataset primarily comprises categorical routing decisions, the branching structure of DTs enables them to efficiently capture and apply decision rules. This suitability explains their superior classification accuracy and underscores the importance of aligning model selection with the specific characteristics of the problem domain, especially in practical routing applications.

To further support our analysis, we reference recent works that explore ML-based routing approaches. Cárdenas et al. [121] introduced GraTree, a routing protocol for vehicular ad hoc networks built on gradient boosting DTs, showcasing the model's effectiveness in managing complex, multi-metric routing decisions. In a similar vein, Kannimuthu et al. [122] proposed DTTrust, a DT-based authentication framework designed to enhance the security of RPL in battlefield IoT environments. Both studies emphasise the practical strengths of DTs in network applications. Their results align with our findings, confirming that DTs are particularly well-suited for routing tasks due to their structured decision logic, flexibility, and computational efficiency, qualities that position them as a compelling choice for ML-driven routing in IoT networks.

4.6.2 Protocols performance comparison

Table 4.3 provides a comparison between our proposed protocol (MLBP) and the AODV protocol in terms of the number of packets sent and the performance gain. The values are derived using the previously defined formulas, based on a scenario involving $N = 105$ messages transmitted over a simulated network comprising $n = 31$ nodes, with an average hop count of $m = \sqrt{32} \approx 6$:

$$sp_{MLBR} = N \times (49 - 43 \times P) \tag{4.11}$$

$$sp_{AODV} = 43 \times N \tag{4.12}$$

$$Gain = N \times (43 \times P - 6) \tag{4.13}$$

Although it would be ideal to compare MLBRP with other ML-based routing approaches, no prior work has addressed routing in IoT networks using historical routing decisions through ML in the specific manner proposed here. Existing studies differ significantly in terms of network settings, datasets, and methodological frameworks, which makes direct comparisons infeasible. As a widely accepted standard, AODV serves as a representative of traditional

routing approaches. Therefore, using it as a benchmark enables a meaningful evaluation of the performance improvements achieved by MLBRP under controlled conditions.

Table 4.3: The sent and gained packets using AODV protocol and MLBRP protocol

	AODV	DT	ANN	SVM(g)	SVM(c)
Accuracy (p)	/	0.87	0.71	0.77	0.83
Number of sent packets	4515	1217	1939	1668	1398
Number of gained packets	0	3298	2576	2847	3117
Percentage of gained packets	0%	73%	57%	63%	69%

Figure 4.13 illustrates how the number of sent packets (sp) varies with the number of transmitted messages when comparing the proposed MLBRP protocol with AODV.

Figure 4.13: Influence of the transmitted messages N on the number of sent packets ($n = 31, m = 6$)

As illustrated in Figure 4.13, our protocol exhibits a slower increase in the number of sent packets compared to AODV as the number of transmitted messages rises. Consequently, the reduction in sent packet overhead achieved by our protocol grows more significant with increased message volume. This indicates that our approach is more efficient under higher network load conditions, as further demonstrated in Figure 4.14:

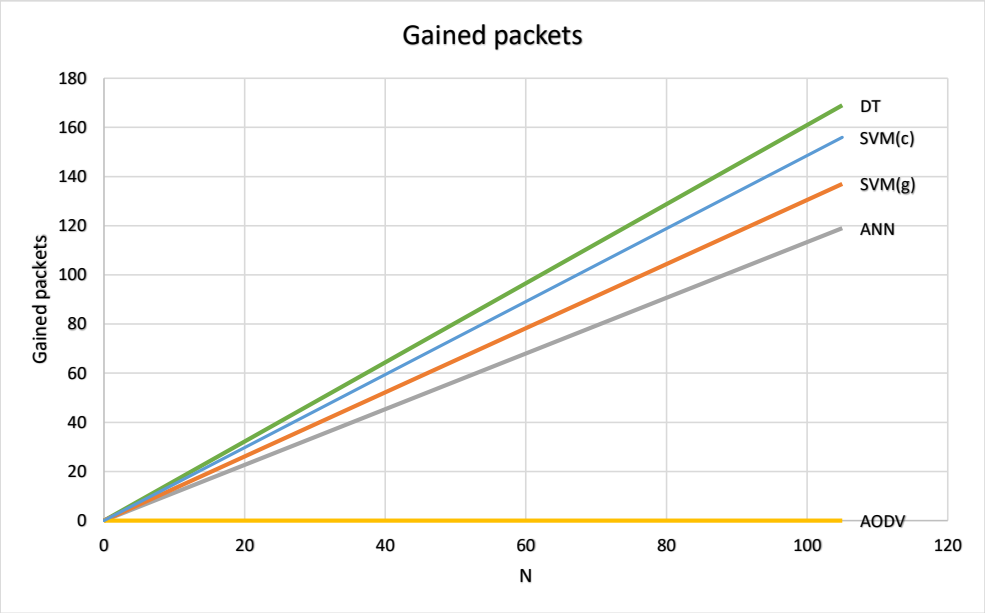

Figure 4.14: Gained packets ($n = 31, m = 6$)

The 3D surface shown in Figure 4.15 represents the variation of the number of sent packets sp as a function of the number of transmitted messages N and the prediction accuracy P of the employed ML model.

As depicted in Figure 4.15, increasing the accuracy of the ML model leads to a noticeable decrease in the number of sent packets needed to deliver a given number of messages. Therefore, achieving high model accuracy is crucial for enhancing performance and optimising packet transmission, as shown in Figure 4.16.

4.6.3 Key performance insights

The proposed Machine Learning-Based Routing Protocol (MLBRP) demonstrates superior performance compared to the traditional AODV protocol across several critical metrics, underscoring its effectiveness for intelligent packet routing in IoT networks. The following performance insights are derived from the obtained experimental results:

a) **Reduced network load:** As depicted in Figure 4.13, MLBRP significantly decreases the volume of sent packets relative to AODV. This reduction is primarily due to the elimination of route discovery and maintenance processes, which are integral to AODV's operation. Furthermore, Figure 4.15 illustrates that the number of transmitted packets continues to decline as the accuracy of the employed ML model increases, emphasising the value of high-accuracy models in optimising routing efficiency.

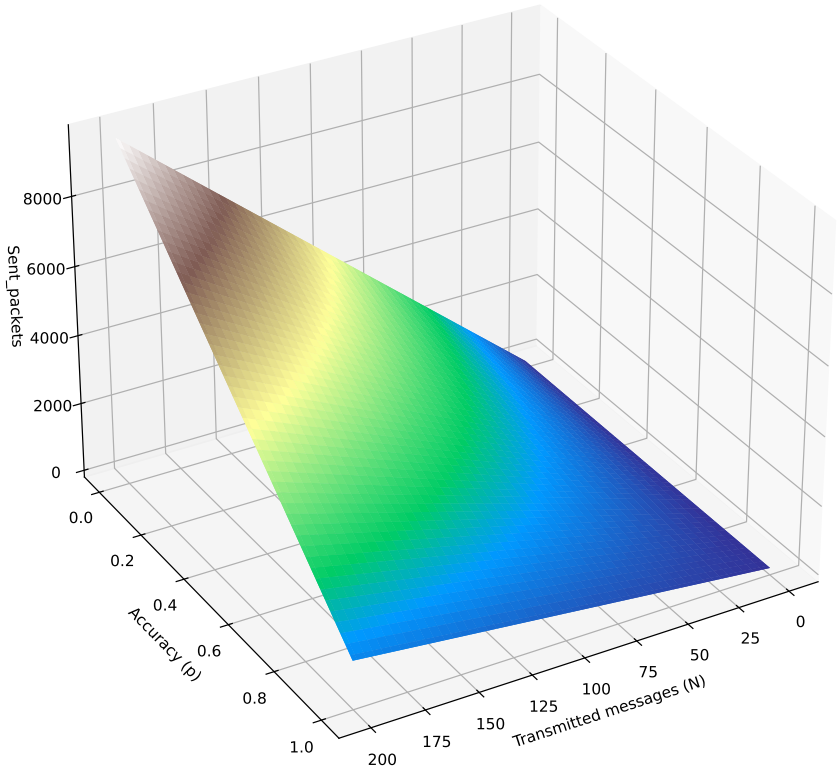

Figure 4.15: Influence of N and P on the number of sent packets

b) **Enhanced packet delivery efficiency:** Figures 4.14 and 4.16 demonstrate that MLBRP yields a higher number of successfully delivered packets compared to AODV. This improvement is attributed to the protocol's ability to predict the optimal next-hop node more accurately, thereby minimising packet loss and unnecessary retransmissions. Additionally, the observed packet gain increases proportionally with the number of transmitted messages, indicating that MLBRP scales effectively with network demand.

c) **Improved energy efficiency:** MLBRP contributes to significant energy savings by reducing both the number of sent and received packets. This is particularly advantageous in IoT contexts, where devices often operate under strict energy constraints. In contrast to AODV's reliance on flooding-based route discovery mechanisms, MLBRP avoids such overhead, thereby prolonging node lifetime and supporting more sustainable network operation.

d) **Reduced packet latency:** As shown in Figure 4.13, MLBRP achieves lower end-to-end delay by bypassing the route discovery process, unlike AODV, which incurs additional latency due to the exchange of RREQ and RREP messages. The latency

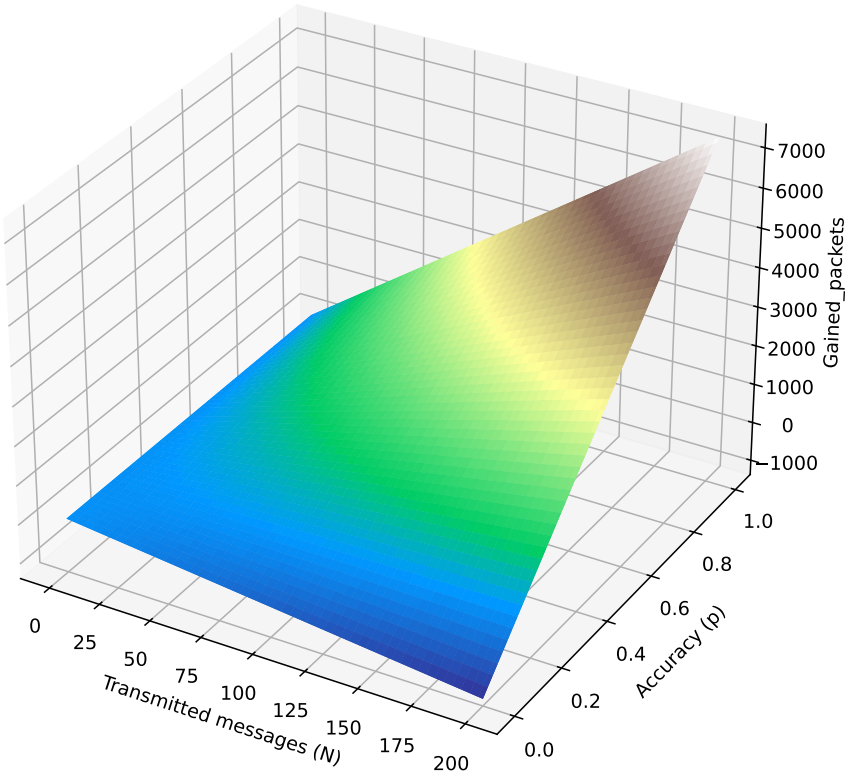

Figure 4.16: Influence of N and P on the number of gained packets

improvement is approximately proportional to $n+m$, making the protocol particularly suitable for latency-sensitive IoT applications.

While the results confirm that increased model accuracy contributes to a reduction in the number of sent packets, the choice of a ML algorithm should not rely on accuracy alone. Other essential aspects, such as model size and response time, also significantly affect the overall efficiency of ML-driven routing decisions.

- **Model size:** The model size, such as that generated by a DT, reflects the amount of information retained after training and required for inference. For instance, DTs produce hierarchical structures composed of multiple nodes, which must be stored locally within network devices. Consequently, the memory capacity of IoT nodes should be carefully considered during model selection and training, especially in resource-limited environments.

- **Model response time:** The response time of a model, defined as the duration between a decision request and the output, can impact end-to-end packet latency. In the case

of DTs, the response time is influenced by the tree's depth, as the decision process involves traversing the tree from the root to a leaf node. The maximum depth is typically equivalent to the number of features in the training dataset.

4.6.4 Validity and real-world applicability of MLBRP

The proposed MLBRP protocol has been thoroughly validated through extensive simulations across diverse performance metrics, confirming its capacity to reduce control overhead, enhance energy efficiency, and adapt dynamically to evolving network conditions. These strengths make it particularly suitable for deployment in resource-constrained IoT environments. The results substantiate the practical feasibility of MLBRP, which introduces an intelligent and efficient alternative to traditional routing protocols such as AODV. By offering substantial improvements in network load reduction, energy conservation, and latency minimisation, MLBRP emerges as a promising solution for next-generation IoT infrastructures. Ongoing research and real-world experimentation will further enhance its scalability and robustness, supporting its application in a wide range of real-world scenarios.

4.7 Conclusion

This chapter introduced the Machine Learning-Based Routing Protocol (MLBRP), a predictive routing strategy designed to improve efficiency in mobile IoT networks. By exploiting repeated mobility patterns typical of smart city environments, MLBRP predicts the next hop without relying on traditional neighbour-based communication or route discovery. Instead, it leverages historical routing data to train supervised ML models capable of making real-time forwarding decisions.

Through extensive simulation, MLBRP demonstrated significant improvements over the AODV protocol in terms of packet delivery, reduced control overhead, energy efficiency, and latency. The results show that as the accuracy of the learning model increases, the number of unnecessary transmissions decreases, highlighting the importance of selecting and training high-precision models.

Nonetheless, the practical deployment of MLBRP must take into account certain limitations. The memory footprint of the trained models and their computational response time can affect performance, particularly on resource-constrained IoT devices. Future enhancements could focus on optimising model complexity, reducing inference latency, and adapting to devices with limited processing capabilities.

This contribution lays the groundwork for more advanced, learning-driven routing mechanisms. The next chapter builds on this foundation by introducing a context-aware DL approach, specifically, a CNN-based framework, that further refines routing predictions using spatial and temporal patterns in dynamic IoT environments.

Chapter 5

Intelligent Packet Routing in IoT Networks Using a CNN-Based Deep Learning Approach

5.1 Introduction

Building on the results of the ML-based routing protocol introduced in the previous chapter, this contribution explores a DL approach to further enhance routing performance in dynamic IoT environments. While classical ML algorithms offer effective decision-making based on predefined features, they often struggle to capture complex spatial and temporal dependencies inherent in mobile networks.

This chapter presents a CNN-based DL routing framework that leverages contextual, topological, and mobility-related information to make more accurate and adaptive forwarding decisions. The proposed model is designed to operate in environments where node movement and interactions follow regular patterns, such as smart city scenarios.

By mapping local observations into structured input representations, the CNN is trained to predict the next optimal hop without requiring neighbour communication or traditional route discovery. This data-driven strategy enhances scalability and reduces overhead, while improving responsiveness to dynamic network conditions.

The chapter details the input encoding strategy, CNN architecture, training methodology, and evaluation setup. Performance is assessed through simulations and compared to both conventional routing protocols and the ML-based method introduced earlier.

5.2 Problem statement

In MANETs within the IoT landscape, conventional routing protocols often exhibit limited adaptability and lack intelligent decision-making capabilities. These protocols typically operate based on static rules and reactive approaches, resulting in suboptimal resource utilisation and network inefficiencies. In particular, they rely heavily on frequent control packet broadcasts for route discovery and periodic HELLO messages for route maintenance, which contribute to significant control overhead and increased communication latency.

In smart city scenarios, populated by heterogeneous mobile nodes such as autonomous vehicles, surveillance systems, and various IoT sensors, mobility patterns are often repetitive and predictable. However, traditional routing protocols treat each routing decision as an isolated event, failing to exploit these recurring patterns. This reactive approach overlooks valuable opportunities to improve routing efficiency through the reuse of previously successful routing experiences.

For instance, consider a typical urban setting where mobile nodes exhibit regular movement routines: a patient visits a clinic twice a week, a student travels to school daily, a teacher commutes to the university several times a week, and a merchant delivers goods regularly. Although their routes differ, these nodes frequently come within each other's communication range, creating temporary yet predictable transmission windows. Traditional routing mechanisms do not take advantage of such recurring encounters, instead initiating route discovery processes from scratch each time. Conversely, an intelligent routing approach that learns from historical communication successes can anticipate future opportunities, enabling proactive and efficient path selection while minimising control message overhead.

This research proposes leveraging DL techniques to analyse historical data on successful packet deliveries and mobility behaviours of diverse nodes. By recognising and learning from these patterns, the objective is to design intelligent routing strategies capable of predicting optimal communication paths based on past performance. This predictive capability aims to enhance routing efficiency, reduce control overhead, and improve overall network performance in MANETs operating within IoT environments such as smart cities.

In conclusion, the communication behaviours and mobility patterns of nodes in smart environments are often cyclical, resulting in predictable opportunities for efficient data transmission. Traditional protocols overlook these patterns, relying instead on repetitive control signalling. Through the application of DL, the proposed approach seeks to identify and utilise these regularities, enabling intelligent next-hop selection based on historical data. This paradigm shift toward a learning-based routing model significantly improves routing perfor-

mance, lowers overhead, and enhances network efficiency in dynamic IoT-driven MANETs.

5.3 Proposed System for Intelligent Routing

This section introduces a DL-based routing system intended to replace conventional routing protocols with a more intelligent, adaptive approach, as illustrated in Figure 5.1.

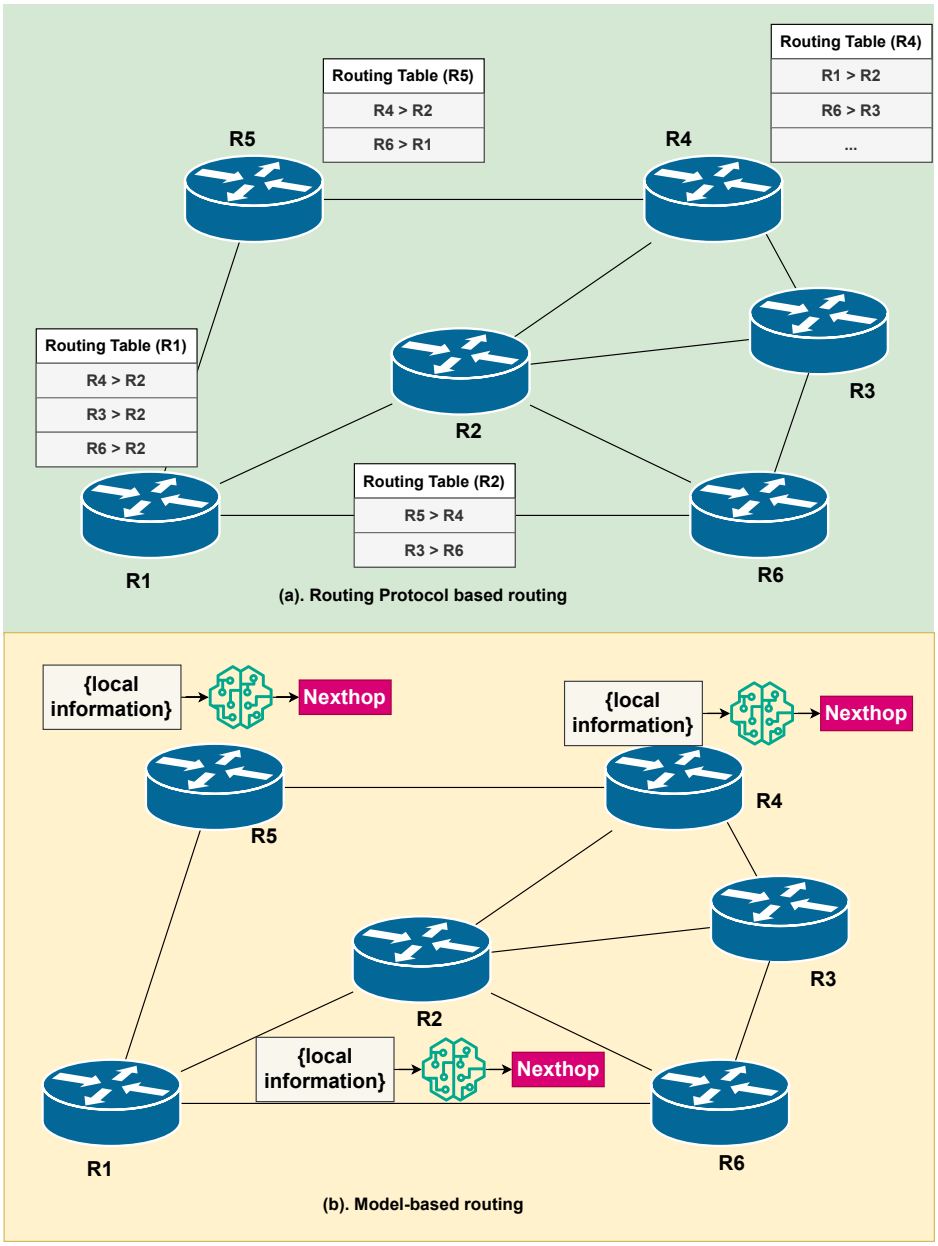

Figure 5.1: Comparison between traditional routing protocols and model-based routing

Unlike traditional methods that rely heavily on frequent control messages for route discovery and maintenance, the proposed system enables network nodes to make autonomous routing decisions based on historical packet transmission data. This reduces the need for continuous interaction with neighbouring nodes during the next-hop selection process.

Conventional routing methods typically depend on frequent exchanges of control messages for route discovery and maintenance, which leads to inefficiencies and fails to utilise insights from prior successful transmissions. To overcome these limitations, our system incorporates a learning mechanism that enables nodes to adapt to dynamic network conditions by leveraging their historical routing experiences.

Recognising the predictable mobility patterns common among nodes in IoT environments, our approach applies DL, specifically CNNs,to predict the most efficient next-hop node for packet forwarding. This predictive capability allows the network to operate with reduced control overhead while maintaining high routing performance.

The proposed system is designed with two primary objectives:

To allow nodes to intelligently select the optimal next-hop from their set of neighbours without the need to initiate a route request for every transmission, thereby minimising communication overhead during the route discovery process.

To ensure adaptability across different network environments and routing protocols, by using any existing protocol as a baseline or "teacher" for the learning model. This hybrid approach enables the model to inherit foundational routing behaviours while enhancing them through learned experience.

Ultimately, our goal is to transition from traditional, rule-based routing to a fully intelligent, data-driven approach that offers improved efficiency, adaptability, and scalability in dynamic IoT-based MANET environments.

As depicted in Figure 5.2, the proposed architecture comprises three core phases:

Data Collection and Preprocessing, DL Model Training, and Intelligent Routing.

5.3.1 Data collection and preprocessing

The development of an effective DL model for intelligent routing requires the careful collection and preprocessing of relevant network data. This process involves the following key steps:

- **Data acquisition:** Routing data is gathered by observing network behaviour over a defined period. Each node functions under a specified routing protocol to make forwarding decisions and determine optimal paths for data transmission. Throughout this phase, nodes record detailed logs of packet forwarding events, including routing

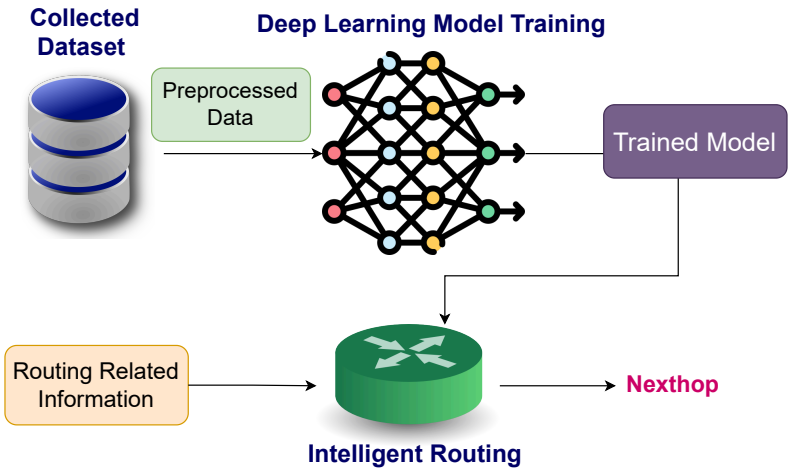

Figure 5.2: Architecture of the Proposed System

decisions and the corresponding network conditions. These logs are systematically stored in structured formats to form a comprehensive dataset suitable for training the model.

- **Feature extraction and data labelling :** Essential features are extracted from the raw data to serve as inputs to the DL model. These features include temporal elements such as date, time, and day, as well as contextual information like the current node's identity and position, neighbouring nodes, and the source and destination of each packet. The dataset is also labelled to differentiate successful routing outcomes. Specifically, each instance is annotated with the next-hop node that successfully delivered the packet, providing a supervised learning signal for model training.

- **Data normalisation :** To enhance model performance and ensure efficient training, the extracted features are normalised to a uniform scale. Normalisation ensures that all input variables contribute proportionally during the learning process, which accelerates convergence and improves overall model accuracy.

5.3.2 DL model training

In this phase, a CNN is designed and trained to predict the most suitable next-hop node based on historical routing data. The training workflow is illustrated in Figure 5.3.

The preprocessed datasets serve as the foundation for building and training the DL models. A distinct CNN model is developed for each participating network node, enabling the

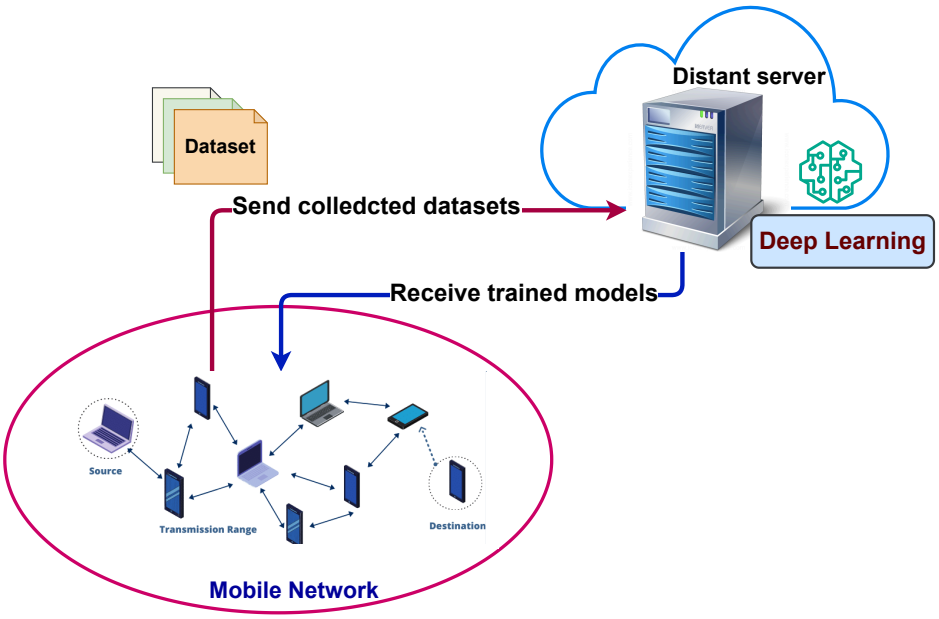

Figure 5.3: Deep learning model training process

node to accurately select the optimal next hop from among its neighbouring nodes.

Given the limited computational capabilities of individual nodes in typical MANET environments, the training process is offloaded to external infrastructure such as servers or cloud platforms. This setup ensures that the computationally intensive tasks, such as training, validation, and hyperparameter tuning, are performed efficiently and without burdening the resource-constrained nodes.

A supervised learning approach is employed, where the CNN is trained to learn patterns and spatial relationships within the routing data. The architecture captures dependencies between node positions, transmission outcomes, and other relevant features. During training, the preprocessed data is input into the network, and model weights are iteratively adjusted through backpropagation. Hyperparameters are fine-tuned to optimise model performance and generalisation.

The goal of this training phase is to equip each node with a predictive model that enhances the accuracy and efficiency of routing decisions. By leveraging learned knowledge from historical data, the system aims to improve overall network performance while minimising control overhead.

5.3.3 Intelligent routing

In this phase, the trained CNN model is deployed within the network to facilitate real-time, intelligent routing. The following steps outline the operational workflow:

- **Model deployment:** The trained CNN model is integrated into the routing infrastructure of the IoT-enabled MANET. It is configured to access real-time information regarding node mobility, network topology, and other contextual features necessary for making accurate routing predictions.
- **Next-hop prediction:** For each data packet, the deployed CNN model analyses the current network state and node-specific attributes to predict the most appropriate next-hop node. This prediction guides the routing process without relying on conventional control message exchanges.
- **Routing decision making:** Based on the model's output, the system selects the next-hop node. If the predicted node is available and satisfies the forwarding criteria (e.g., within transmission range, suitable link quality), the packet is forwarded accordingly. This process eliminates the need for frequent route discovery broadcasts.
- **Continuous learning:** To maintain model relevance and adaptability, a mechanism for incremental learning is introduced. As the network evolves and new data is collected, the CNN model is periodically retrained to refine its prediction accuracy and respond to changing network dynamics.
- **Fallback mechanism:** To ensure reliability, a fallback mechanism is implemented. If the CNN model fails to generate a valid prediction, such as when a packet is undelivered or misrouted, the system reverts to a traditional routing protocol. This hybrid approach enhances robustness, ensuring continued operation even under uncertain conditions.

The entire process is visually represented in Figure 5.4.

Through this intelligent routing mechanism, the proposed system leverages DL to predict optimal communication paths based on historical transmission data and recurring mobility patterns. This approach significantly reduces reliance on traditional routing protocols, minimises control overhead, and enhances the overall efficiency and adaptability of the network in dynamic IoT environments.

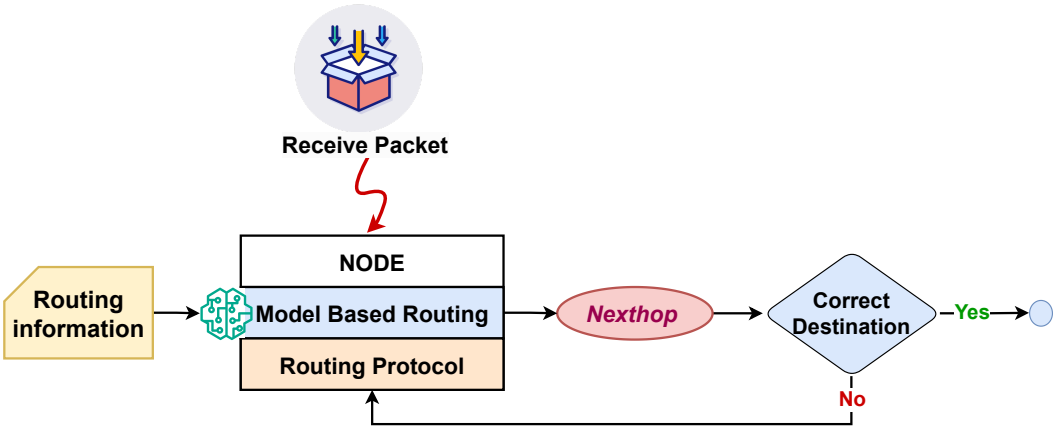

Figure 5.4: Deep learning-based intelligent routing

5.4 Performance evaluation

This section presents a performance analysis of the proposed DL-based routing approach, carried out through simulation experiments using the NS2 network simulator [119]. For benchmarking, we compare the performance of our method against the AODV protocol, a widely used and straightforward routing algorithm in MANETs. While AODV serves as the baseline in our evaluation, it is important to emphasise that our proposed approach is adaptable and can be integrated with other routing protocols as well.

To conduct the simulations, we consider a network scenario consisting of 50 mobile nodes operating over a simulated duration of 500 seconds per day for one month. Each node follows a regular mobility model, wherein nodes periodically come within communication range of one another, simulating the repetitive movement patterns typical in IoT-enabled smart city environments.

Following the configuration of essential network parameters, the simulation is executed. During this process, all relevant information regarding successful packet transmissions and routing decisions is logged. The resulting dataset comprises 20,000 data samples, each representing a routing decision made by a node based on observed network conditions and mobility behaviour.

5.4.1 Training the CNN model

Once the dataset is collected, the next phase involves training the CNN responsible for intelligent routing decision-making.

5.4.1.1 Model architecture

The core of our DL-based routing strategy is a CNN model trained to predict the next-hop node using historical transmission data. The model architecture begins with an input layer that processes a 2×4 matrix of features. These features represent contextual and temporal attributes of the current routing state.

The input is passed through a convolutional layer with a 2×2 kernel, followed by batch normalisation and max pooling to extract relevant patterns and reduce spatial dimensions. To enhance learning efficiency and address the vanishing gradient problem, the architecture incorporates two residual blocks. The first block contains 32 filters, and the second has 64 filters, with each block comprising two convolutional layers, batch normalisation, and a shortcut connection.

Dimensionality is further reduced using max pooling, and dropout layers are introduced to prevent overfitting. The resulting feature maps are flattened and processed through two fully connected (dense) layers with 256 and 128 neurons, respectively. These layers use ReLU activation and L2 regularisation to enhance generalisation and mitigate overfitting.

Finally, the output layer uses softmax activation to generate a probability distribution over the 50 possible next-hop nodes, allowing the model to select the most likely candidate for packet forwarding.

The complete architecture is illustrated in Figure 5.5, which provides a visual representation of the CNN's structure and components.

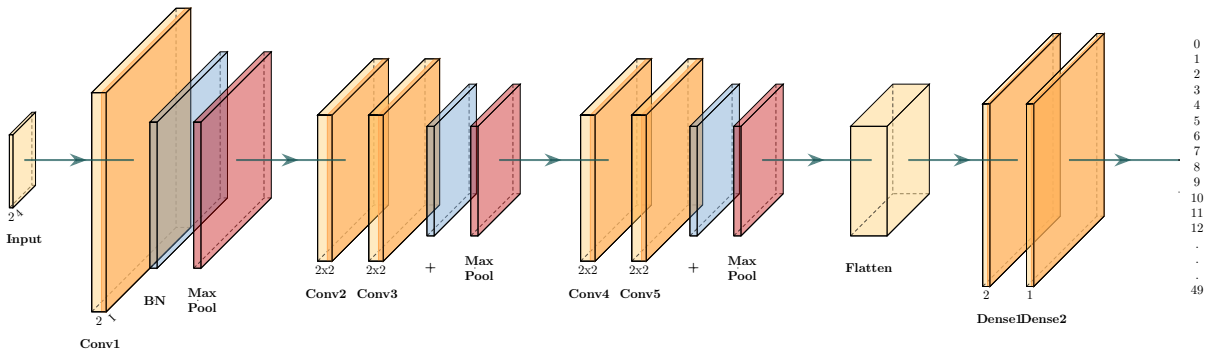

Figure 5.5: CNN architecture for next-hop prediction

5.4.1.2 Training process

The training process begins with thorough data preparation. The collected dataset is first cleaned and normalised to ensure consistency across all features. It is then partitioned into

separate training and testing subsets. The input data is reshaped to align with the input format expected by the CNN, and the output labels are converted into categorical form to support multiclass classification.

The model is compiled using the Adam optimiser, selected for its efficiency and adaptive learning capabilities, along with the categorical cross-entropy loss function, which is well-suited for classification tasks involving multiple classes. The CNN is trained over a maximum of 2000 epochs. During this process, the model progressively adjusts its internal weights to minimise prediction error. Upon completion, the model achieved a training accuracy of 70.29% and a test accuracy of 87.60%, indicating strong generalisation and effective learning from the data.

5.4.2 Model evaluation

Model evaluation is a critical step in supervised learning, aimed at determining how well the trained model performs on previously unseen data. This ensures the model's ability to generalise beyond the training dataset. In this study, we assess performance primarily using the accuracy metric, a straightforward yet powerful measure of classification effectiveness.

Accuracy, denoted as P, quantifies the proportion of correctly predicted instances relative to the total number of test instances. In the context of our routing model, it represents the ratio of data packets correctly routed by the CNN model to the total number of packets transmitted. The accuracy is formally defined as:

$$P = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \tag{5.1}$$

Where:
N is the total number of transmitted packets.
y_i denotes the next-hop node selected by the baseline AODV protocol for packet i.
$f(x_i)$ represents the next-hop node predicted by the CNN model for the same packet.
$L(y_i, f(x_i))$ is a binary indicator function that returns 1 if the prediction is correct ($y_i = f(x_i)$), and 0 otherwise.

This metric provides a direct comparison between our proposed DL-based routing model and the standard AODV protocol. By evaluating how often the CNN model aligns with AODV's routing decisions, which are based on traditional reactive mechanisms, we can measure the predictive accuracy and effectiveness of our approach.

Furthermore, this evaluation offers insight into the model's real-world applicability. A higher accuracy indicates the model's ability to make reliable routing decisions, thereby enhancing overall network performance by reducing latency and control overhead. The results affirm the potential of the proposed model to serve as a viable alternative or supplement to conventional routing protocols in dynamic IoT and MANET environments.

5.4.3 Method evaluation

To assess the efficiency of our proposed CNN-based routing model, we compare its performance against the AODV protocol with a specific focus on the **total number of packets transmitted** during the routing process. This evaluation metric reflects the overhead associated with control message exchanges and data transmission, offering a practical measure of protocol efficiency.

A. Total number of packets transmitted by the AODV protocol

The AODV protocol is a reactive routing strategy designed for highly dynamic wireless environments. Unlike proactive routing protocols, which continuously maintain up-to-date route tables, AODV establishes routes only when required (on demand), thereby reducing idle overhead but incurring setup costs during route discovery.

Before data can be transmitted, a source node checks its routing table for a valid route to the destination. If no such route exists, the node initiates a route discovery process using two primary control messages: **RREQs** and **RREPs**.

a) **Route Request**

The route discovery process begins when the source node broadcasts a *RREQ* message to its neighbouring nodes. These nodes, in turn, rebroadcast the RREQ to their neighbours, propagating the request throughout the network until it either reaches the destination or an intermediate node with a valid route to the destination.

Given a network with n total nodes, the RREQ message may potentially be forwarded up to n times in the worst-case scenario. Although mechanisms such as unique identifiers help reduce redundant forwarding and prevent loops, the approximate number of RREQ packets generated is still proportional to the size of the network:

$$C_{\text{RREQ}} \approx n$$

b) **Route Reply** Once the RREQ reaches the destination or a qualified intermediate node, a *RREP* message is sent back to the source node. This message travels in unicast

mode along the reverse path established during RREQ propagation. If the number of hops between the source and destination is h, then the RREP transmission incurs:

$$C_{\text{RREP}} = h$$

c) **Data packets** Following the successful establishment of the route, data packets can be transmitted along the determined path. Each data packet is forwarded through h intermediate nodes (i.e., over h hops) from the source to the destination. Therefore, for each data transmission, the packet travels across h nodes:

$$C_{\text{DATA}} = h$$

Therefore, the total number of packets transmitted when a node sends a single data packet to another node consists of three components: the total number of broadcasted RREQ control messages, the total number of unicast RREP control messages, and the number of data packets forwarded through the intermediate nodes between the source and destination. This total is denoted as M_{AODV}:

$$\begin{aligned} M_{\text{AODV}} &= C_{\text{RREQ}} + C_{\text{RREP}} + C_{\text{DATA}} \\ &\approx n + h + h = n + 2 \times h \end{aligned} \tag{5.2}$$

Here, n represents the total number of nodes in the network, and h denotes the average number of hops between the source and destination nodes.

Consequently, when transmitting N data packets using the AODV protocol, the overall packet overhead generated across the network is:

$$M_{AODV} = N \times (n + 2 \times h) \tag{5.3}$$

B. Total number of packets transmitted by CNN model

Let P denote the accuracy of our trained DL model in performing routing tasks. This accuracy reflects the fraction of data packets correctly delivered from their sources to intended destinations using our approach. Given N as the total number of data packets transmitted network-wide, the number of successfully routed packets is $P \times N$.

Assuming an average hop count h for each packet, the total number of data packets forwarded for successful transmissions is:

$$M_{\text{success}} = P \times N \times h \tag{5.4}$$

Conversely, $(1-P)\times N$ represents the number of packets that fail to reach their destination via the CNN model. The total data packets forwarded for these unsuccessful transmissions are:

$$M_{\text{failed}} = (1 - P) \times N \times h \tag{5.5}$$

Since failed transmissions require fallback to the AODV protocol for retransmission, the number of retransmitted packets over the network is:

$$\begin{aligned} M_{\text{retransmit}} &= (1 - P) \times N \times M_{AODV} \\ &= (1 - P) \times N \times (n + 2 \times h) \end{aligned} \tag{5.6}$$

Therefore, the total packet overhead incurred by our CNN-based routing method, accounting for successful transmissions, failed attempts, and retransmissions via AODV, is given by:

$$\begin{aligned} M_{\text{RCNN}} &= M_{\text{success}} + M_{\text{failed}} + M_{\text{retransmit}} \\ &= N \times h + (1 - P) \times N \times (n + 2 \times h) \end{aligned} \tag{5.7}$$

5.4.4 Results

A. Model accuracy and loss

To assess the performance of the trained CNN model, both training and testing phases were conducted. The outcomes are illustrated in Figure 5.6, which presents the trends in model accuracy and loss across 2000 training epochs.

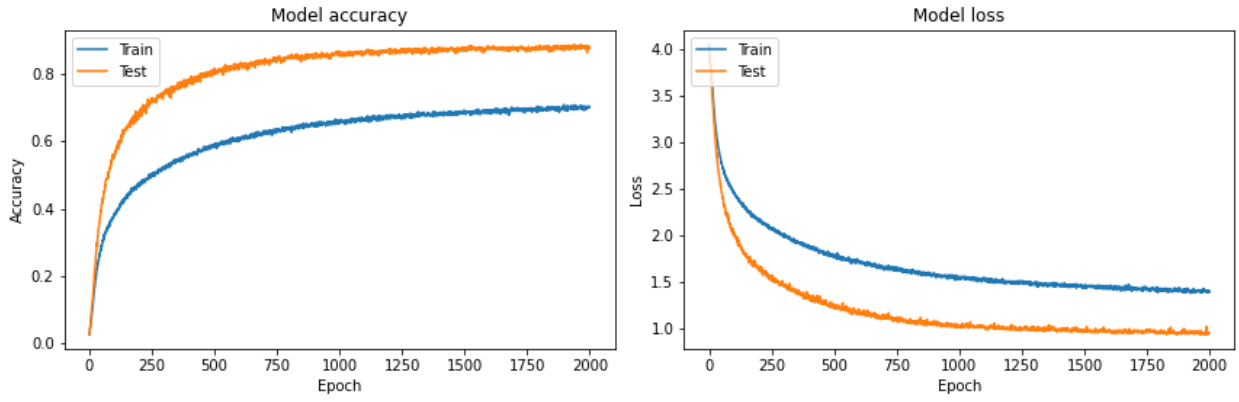

Figure 5.6: CNN model performance: accuracy and loss

a) **Accuracy**

The graph on the left shows the progression of accuracy for both the training and testing datasets throughout the training process. Key insights include:

- **Training accuracy:** Initially low, the training accuracy increases gradually with the number of epochs and stabilises at around 0.65. This trend indicates that the model is successfully learning from the training data, though performance could still be improved.
- **Testing accuracy:** The testing accuracy experiences a rapid rise during the early epochs, surpassing the training accuracy and ultimately reaching a stable level of approximately 0.87. This implies strong generalisation to unseen data and suggests that overfitting is not a concern.

b) **Loss**

The graph on the right illustrates the loss values for both training and testing datasets over the same epoch range. Observations include:

- **Training loss:** Starting from a high value, the training loss consistently declines, reaching approximately 1.5 by the end of the training period. This demonstrates the model's effectiveness in reducing prediction error on the training data.
- **Testing loss:** Similarly, the testing loss decreases throughout the training process. Although it starts higher than the training loss, it eventually drops to about 1.0. This outcome reflects the model's ability to capture meaningful patterns from the data while maintaining resistance to overfitting.

5.4.5 Discussion

The evaluation results presented in the performance graphs highlight the effectiveness of our CNN-based routing model. The key observations are summarised as follows:

- **Strong test accuracy:** Achieving a test accuracy of approximately 87% confirms the model's capability to accurately predict the next hop for data packet forwarding. This high performance on unseen data reflects the model's strong generalisation ability.
- **Reduction in loss:** The steady decline in both training and testing loss demonstrates the model's capacity to efficiently learn from the dataset. The decreasing loss values indicate that the model effectively minimises prediction errors while maintaining robustness against overfitting.

Overall, these outcomes confirm the practical relevance of our CNN-driven routing strategy in dynamic MANET scenarios. Compared to the traditional AODV protocol (as referenced in the performance analysis), our model achieves higher accuracy with lower packet loss, resulting in significantly improved routing decisions. This reinforces the model's potential to enhance routing efficiency and overall network performance.

5.4.5.1 Comparative packet overhead analysis

To assess the efficiency of the proposed CNN-based routing approach (RCNN) in comparison to the AODV protocol, we performed an analytical evaluation using a network consisting of $n = 50$ nodes. The average hop count is estimated as $h = \sqrt{n} = \sqrt{50} \approx 7$, and the model's accuracy is set at $P = 0.87$. Based on these parameters, the total number of transmitted packets for each protocol is determined using the following expressions:

$$M_{\text{RCNN}} = N \times h + (1 - P) \times N \times (n + 2 \times h) \tag{5.8}$$

$$M_{\text{AODV}} = N \times (n + 2 \times h) \tag{5.9}$$

Upon simplification, the equations become:

$$M_{\text{RCNN}} = N \times (71 - 64 \times P) = 15 \times N \tag{5.10}$$

$$M_{\text{AODV}} = 64 \times N \tag{5.11}$$

Figure 5.7 illustrates the relationship between the number of sent packets (SP) and the total number of transmitted messages (N) for both the RCNN and AODV protocols.

As shown in the figure, the number of packets sent by the AODV protocol grows significantly with the increase in transmitted messages. In contrast, the RCNN approach achieves notably greater efficiency, requiring substantially fewer packets to deliver the same number of messages. This reinforces the RCNN protocol's advantage in reducing routing overhead and improving scalability in dynamic network environments.

5.4.5.2 Performance improvements of RCNN over AODV

The proposed RCNN protocol exhibits clear performance improvements when compared to the conventional AODV protocol across several important network metrics:

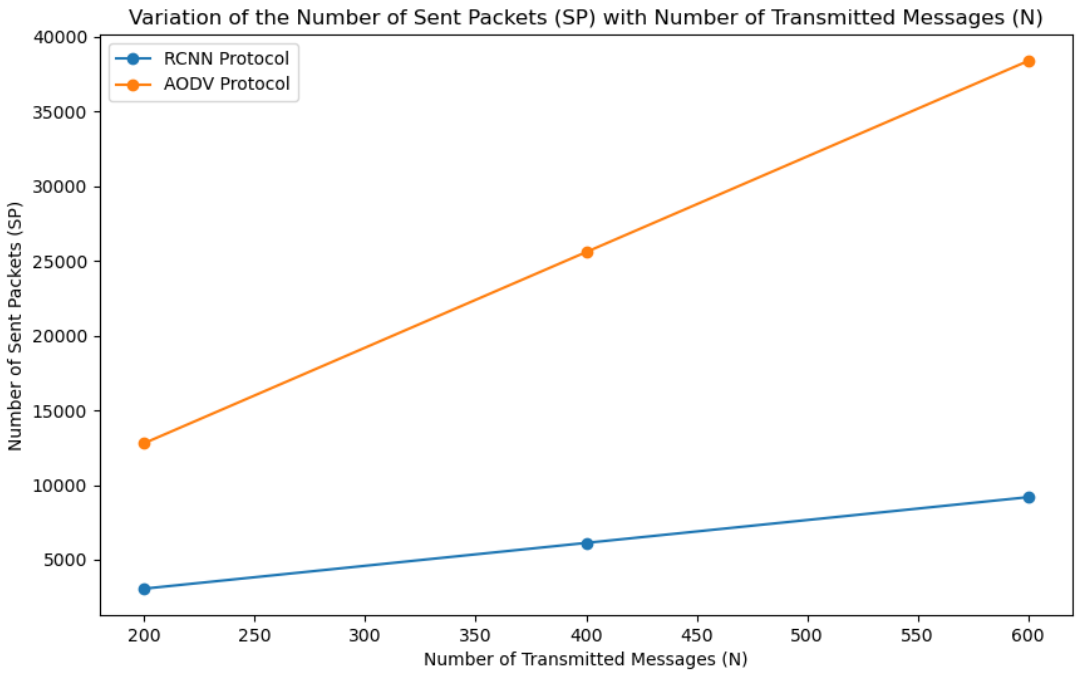

Figure 5.7: Comparison of Sent Packets: RCNN vs. AODV Protocols with Varying Transmitted Messages (N)

- **Packet transmission efficiency:** As the number of transmitted messages (N) increases, AODV experiences a rapid rise in the number of sent packets (SP), reaching up to 35,000 packets for 600 messages. In contrast, the RCNN protocol maintains a significantly lower and more stable packet count, reflecting its higher transmission efficiency.
- **Reduction in control overhead:** Both the analytical derivations and visual results confirm that RCNN generates substantially fewer control messages than AODV. This reduction lowers routing overhead and contributes to a more efficient network operation, especially in dynamic environments.
- **Lower end-to-end delay:** Unlike AODV, which relies on route discovery, RCNN bypasses this phase by leveraging a trained model to make routing decisions instantly. This results in reduced delivery time and avoids delays due to congestion, ensuring better real-time performance.
- **Energy efficiency:** By minimising the number of sent packets and avoiding unnecessary retransmissions, RCNN conserves more energy. This is especially beneficial in

MANETs, where devices are typically battery-powered. Reduced energy consumption extends node lifetime and supports longer network operation.

- **Enhanced throughput and reduced routing overhead:** RCNN achieves higher throughput by forwarding more data with fewer transmissions. The lower generation of control messages also minimises routing overhead, enabling more efficient use of bandwidth and computational resources. This makes RCNN suitable for dense or highly mobile network scenarios where performance and resource efficiency are critical.

These findings highlight the robustness and practicality of the RCNN approach for routing in MANETs. Its ability to improve data delivery accuracy, reduce overhead, and optimise energy and delay metrics demonstrates its potential for deployment in dynamic and resource-constrained network environments.

5.5 Conclusion

This chapter introduced a CNN-based routing framework for mobile IoT networks, designed to predict the next hop by learning spatial and contextual patterns from local observations. Unlike traditional protocols and classical ML models, the proposed approach eliminates the need for neighbour communication or route discovery.

Simulation results show that the CNN model outperforms both AODV and the previously proposed MLBRP in terms of packet delivery, latency, energy efficiency, and control overhead. These gains highlight the strength of DL in capturing complex dependencies within dynamic environments.

Despite its advantages, the approach raises challenges related to model complexity and deployment on resource-limited devices. Future work may explore model compression or lightweight inference techniques to improve practicality.

This contribution confirms the potential of DL for intelligent, adaptive routing in IoT networks.

General conclusion

1 Summary

IoT is reshaping modern digital infrastructure by enabling efficient interaction between physical objects and computational systems, but its success heavily relies on the robustness of underlying data communication mechanisms, particularly routing. In smart cities, the deployment of IoT devices offers numerous advantages across transportation, environmental monitoring, public safety, and urban services, provided that the network communication remains efficient and adaptive.

However, the effectiveness of such systems critically depends on intelligent and adaptive packet routing strategies suited for highly dynamic and resource-constrained environments.

This thesis addresses the fundamental challenges of routing in mobile IoT networks, where traditional approaches like AODV fall short due to their static design, heavy control overhead, and limited adaptability. These limitations, including inefficiency, lack of learning capability, and failure to leverage mobility patterns, underscore the need for routing mechanisms that are context-aware, predictive, and scalable.

To respond to these challenges, two major contributions were proposed and validated:

A. Machine Learning-Based Routing Protocol (MLBRP)

The first contribution introduced a novel routing protocol grounded in classical machine learning (ML) techniques. In MLBRP, each node is equipped with a lightweight, pre-trained model that leverages knowledge extracted from previous routing decisions and local contextual features. This approach marks a shift from traditional rule-based routing to predictive, experience-informed decision-making.

This protocol was designed to:

- Minimise the overhead caused by frequent control message broadcasts.

- Exploit repetitive mobility patterns found in urban environments to anticipate efficient paths.
- Improve energy efficiency and scalability through context-aware, intelligent decision-making.

The MLBRP was evaluated using DTs, SVMs, and ANN, each offering varying trade-offs between accuracy and computational complexity. Simulation results demonstrated that MLBRP significantly outperforms conventional approaches by reducing routing overhead and conserving energy, particularly in dynamic topologies typical of smart cities.

B. CNN-based deep learning routing approach

Building upon the MLBRP foundation, the second contribution advanced the routing process through a deep learning approach using CNNs. This framework was designed to automatically learn complex spatial and contextual patterns from a rich input feature set, allowing for precise next-hop prediction without the need for control packet exchange.

This contribution involved:

- Constructing a comprehensive dataset that integrates contextual, temporal, and position features.
- Designing a CNN architecture with convolutional and dense layers to extract relevant patterns from observed data.
- Integrating smart city mobility regularities into the training process to improve model relevance.
- Demonstrating superior routing reliability and accuracy over both traditional and classical ML approaches.

The CNN-based model achieved more robust performance under challenging conditions, such as dense traffic and frequent topology changes. It showed that deep learning can significantly improve routing decisions by enabling adaptive, context-aware intelligence at each node, even in resource-constrained environments.

C. Research impact

Through these two contributions, this thesis introduces and validates a paradigm shift in packet routing for IoT networks, transitioning from conventional, rule-based routing approaches to intelligent, data-driven protocols. The proposed models reduce network overhead,

lower energy consumption, and provide greater adaptiveness and autonomy that aligns with the complex, dynamic nature of smart city infrastructures.

By leveraging predictable mobility patterns and the capabilities of ML and DL, this work demonstrates the feasibility of routing systems that learn from experience, reason based on context, and adapt to future changes without requiring continuous reconfiguration or human intervention.

2 Perspectives and future work

Although the results achieved are promising, several directions remain open for further research:

- **Real-world deployment:** Testing and deploying the proposed models on real IoT platforms and smart city infrastructures would provide valuable insights into their practical applicability, robustness, and limitations.
- **Online learning and adaptation:** Future models could incorporate online or continual learning mechanisms to dynamically adjust to new patterns, network anomalies, or environmental changes.
- **Model compression and edge deployment:** Optimizing CNN models for low-power IoT devices through techniques such as pruning, quantization, or knowledge distillation will facilitate deployment in real-world, constrained devices.
- **Security and trust integration:** Integrating trust management or anomaly detection into the learning-based routing framework can improve resilience against malicious nodes or routing attacks.
- **Multi-objective routing:** Future frameworks could consider additional criteria, such as latency, reliability, and quality-of-service constraints, to support more diverse application requirements in smart cities.

In conclusion, this thesis advances the field of smart networking by proposing practical and intelligent routing models specifically designed for mobile IoT environments. It lays a solid foundation for future research on autonomous, adaptive, and learning-driven communication systems capable of addressing the dynamic demands of next-generation smart cities.

Bibliography

[1] Andrew Whitmore, Anurag Agarwal, and Li Da Xu. "The Internet of Things—A survey of topics and trends". In: *Information Systems Frontiers* 17.2 (2015), pp. 261–274. DOI: 10.1007/s10796-014-9489-2 (cit. on p. 1).

[2] Andrea Zanella et al. "Internet of Things for Smart Cities". In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 22–32. DOI: 10.1109/JIOT.2014.2306328 (cit. on p. 1).

[3] "Internet of things: Vision, applications and research challenges". In: *Ad Hoc Networks* 10.7 (2012), pp. 1497–1516. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2012.02.016 (cit. on p. 1).

[4] N. N. Srinidhi, S. M. Dilip Kumar, and K. R. Venugopal. "Network Optimizations in the Internet of Things: A Review". In: *Engineering Science and Technology, an International Journal* 22.1 (2019), pp. 1–21. DOI: 10.1016/j.jestch.2018.09.003 (cit. on pp. 1, 23).

[5] M. Ersue et al. *Management of Networks with Constrained Devices: Problem Statement and Requirements*. RFC: 7547. 2015. URL: https://www.rfc-editor.org/rfc/rfc7547.html (cit. on p. 1).

[6] K. Kabilan et al. "Performance Analysis of IoT Protocol Under Different Mobility Models". In: *Computers & Electrical Engineering* 72 (2018), pp. 154–168. DOI: 10.1016/j.compeleceng.2018.09.007 (cit. on p. 2).

[7] Saqib Daud et al. "DSDV and AODV Protocols Performance in Internet of Things Environment". In: *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*. 2019, pp. 466–470. DOI: 10.1109/ICCSN.2019.8905256 (cit. on p. 2).

[8] Vu Khanh Quy et al. "Routing Algorithms for MANET-IoT Networks: A Comprehensive Survey". In: *Wireless Personal Communications* 125.4 (2022), pp. 3501–3525. ISSN: 1572-834X. DOI: 10.1007/s11277-022-09722-x (cit. on p. 2).

[9] Amira Zrelli, Hacen Khlaifi, and Tahar Ezzedine. "Performance Evaluation of AODV and OAODV for Several WSN/IoT Applications". In: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2019, pp. 1–6. DOI: 10.23919/SOFTCOM.2019.8903830 (cit. on p. 2).

[10] H. Boukhedouma et al. "On the Challenges of Mobility Prediction in Smart Cities". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIV-4/W2-2020 (2020), pp. 17–24. DOI: 10.5194/isprs-archives-XLIV-4-W2-2020-17-2020 (cit. on p. 2).

[11] William Su, S.-J. Lee, and Mario Gerla. "Mobility Prediction in Wireless Networks". In: *MILCOM 2000 Proceedings: 21st Century Military Communications—Architectures and Technologies for Information Superiority*. Vol. 1. 2000, pp. 491–495. DOI: 10.1109/MILCOM.2000.905001 (cit. on p. 2).

[12] Yingzi Wang et al. "Regularity and Conformity: Location Prediction Using Heterogeneous Mobility Data". In: *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 1275–1284. DOI: 10.1145/2783258.2783350 (cit. on p. 2).

[13] 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), p. 16. DOI: 10.1186/s13174-018-0087-2 (cit. on pp. 2, 34, 40).

[14] Yuyi Mao et al. "A Survey on Mobile Edge Computing: The Communication Perspective". In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2322–2358. DOI: 10.1109/COMST.2017.2745201 (cit. on p. 2).

[15] Zhi Zhou et al. "Edge intelligence: Paving the last mile of artificial intelligence with edge computing". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1738–1762. DOI: 10.1109/JPROC.2019.2918951 (cit. on p. 2).

[16] Raafi Careem, G. Johar, and Ali Khatibi. "Deep Neural Networks Optimization for Resource-Constrained Environments: Techniques and Models". In: *Indonesian Journal of Electrical Engineering and Computer Science* 33.3 (2024), pp. 1843–1854. ISSN: 2502-4752. DOI: 10.11591/ijeecs.v33.i3.pp1843-1854 (cit. on p. 2).

[17] Fengxiao Tang et al. "On Removing Routing Protocol from Future Wireless Networks: A Real-time Deep Learning Approach for Intelligent Traffic Control". In: *IEEE Wireless Communications* 25.1 (2018), pp. 154–160. DOI: 10.1109/MWC.2017.1700244 (cit. on pp. 2, 65).

[18] Yushan Jiang et al. "Spatial–Temporal Graph Data Mining for IoT-Enabled Air Mobility Prediction". In: *IEEE Internet of Things Journal* 9.12 (2022), pp. 9232–9240. DOI: 10.1109/JIOT.2021.3090265 (cit. on p. 2).

[19] Lingwei Xu et al. "Performance Analysis and Prediction for Mobile Internet-of-Things (IoT) Networks: A CNN Approach". In: *IEEE Internet of Things Journal* 8.17 (2021), pp. 13355–13366. DOI: 10.1109/JIOT.2021.3065368 (cit. on p. 2).

[20] Daniel Minoli. *Building the Internet of Things with IPv6 and MIPv6: The Evolving World of M2M Communications*. John Wiley & Sons, 2013 (cit. on p. 7).

[21] Aidan Fuller et al. "Digital twin: enabling technologies, challenges and open research". In: *IEEE access* 8 (2020), pp. 108952–108971. DOI: 10.1109/ACCESS.2020.2998358 (cit. on p. 8).

[22] Ala Al-Fuqaha et al. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376. DOI: 10.1109/COMST.2015.2444095 (cit. on pp. 8, 13).

[23] Muhammad Burhan et al. "IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey". In: *Sensors* 18.9 (2018), p. 2796. DOI: 10.3390/s18092796 (cit. on pp. 10, 13–15).

[24] Noboru Koshizuka and Ken Sakamura. "Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things". In: *IEEE Pervasive Computing* 9.4 (2010), pp. 98–101. DOI: 10.1109/MPRV.2010.87 (cit. on p. 10).

[25] Gabriel Montenegro et al. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944. IETF, 2007. URL: https://datatracker.ietf.org/doc/html/rfc4944 (cit. on p. 10).

[26] Nandakishore Kushalnagar, Gabriel Montenegro, and Christian Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919. IETF, 2007. URL: https://datatracker.ietf.org/doc/html/rfc4919 (cit. on p. 10).

[27] Ian F. Akyildiz et al. "A Survey on Sensor Networks". In: *IEEE Communications Magazine* 40.8 (2002), pp. 102–114. DOI: 10.1109/MCOM.2002.1024422 (cit. on p. 10).

[28] Mustafa Jamal Mahdi, Abbas Fadhil Aljuboori, and A. M. Hussein. "Smart Stadium Using Cloud Computing and Internet of Things (IoT): Existing and New Models". In: *International Journal of Computer Applications Technology and Research* 10.5 (2021), pp. 111–118 (cit. on p. 13).

[29] Pallavi Sethi and Smruti R. Sarangi. "Internet of Things: Architectures, Protocols, and Applications". In: *Journal of Electrical and Computer Engineering* 2017 (2017). DOI: 10.1155/2017/9324035 (cit. on pp. 13, 16).

[30] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A Survey". In: *Computer Networks* 54.15 (2010), pp. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010 (cit. on pp. 15, 17).

[31] Mussab Alaa et al. "A review of smart home applications based on Internet of Things". In: *Journal of Network and Computer Applications* 97 (2017), pp. 48–65. DOI: 10.1016/j.jnca.2017.08.017 (cit. on p. 16).

[32] Nuno Vasco Moreira Lopes. *Smart Governance for Cities: Perspectives and Experiences*. Springer, 2020 (cit. on p. 19).

[33] Md. Whaiduzzaman et al. "A Review of Emerging Technologies for IoT-Based Smart Cities". In: *Sensors* 22.23 (2022), p. 9271. ISSN: 1424-8220. DOI: 10.3390/s22239271 (cit. on p. 20).

[34] Cuong Duc Truong. "Routing and Sensor Search in the Internet of Things". PhD thesis. University of Lübeck, 2014. URL: https://www.zhb.uni-luebeck.de/epubs/ediss1361.pdf (cit. on pp. 20, 21, 24, 29).

[35] Sukanta Dey, Pradeepkumar Bhale, and Sukumar Nandi. "ReFIT: Reliability Challenges and Failure Rate Mitigation Techniques for IoT Systems". In: *International Conference on Innovations for Community Services*. Springer, 2019, pp. 123–142. DOI: 10.1007/978-3-030-37484-6_7 (cit. on p. 22).

[36] Patrick Olivier Kamgueu. "Configuration Dynamique et Routage pour l'Internet des Objets". PhD thesis. Université de Lorraine, 2017. URL: https://theses.hal.science/tel-01687704/ (cit. on pp. 22, 24, 29–31).

[37] Jamal N. Al-Karaki and Ahmed E. Kamal. "Routing Techniques in Wireless Sensor Networks: A Survey". In: *IEEE Wireless Communications* 11.6 (2004), pp. 6–28. DOI: 10.1109/MWC.2004.1368893 (cit. on p. 25).

[38] Ravi Kumar Poluru and Shaik Naseera. "A Literature Review on Routing Strategy in the Internet of Things". In: *Journal of Engineering Science and Technology Review* 10.5 (2017) (cit. on p. 27).

[39] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. 6th ed. Addison-Wesley, 2013. ISBN: 978-0132856201 (cit. on p. 28).

[40] Jithin Jagannath et al. "Machine learning for wireless communications in the Internet of Things: A comprehensive survey". In: *Ad Hoc Networks* 93 (2019), p. 101913. DOI: 10.1016/j.adhoc.2019.101913 (cit. on p. 32).

[41] Rejab Hajlaoui et al. "Protecting Machine Learning Systems Using Blockchain: Solutions, Challenges and Future Prospects". In: *Multimedia Tools and Applications* (2024), pp. 1–28. DOI: 10.1007/s11042-024-19993-0 (cit. on p. 34).

[42] Issam El Naqa and Martin J. Murphy. *What is Machine Learning?* Springer, 2015 (cit. on p. 33).

[43] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210 (cit. on p. 33).

[44] Ethem Alpaydın. *Introduction to Machine Learning*. 2nd ed. MIT Press, 2011 (cit. on p. 34).

[45] Khadija El Bouchefry and Rafael S. de Souza. "Chapter 12 - Learning in Big Data: Introduction to Machine Learning". In: *Knowledge Discovery in Big Data from Astronomy and Earth Observation*. Ed. by Petr Škoda and Fathalrahman Adam. Elsevier, 2020, pp. 225–249. DOI: 10.1016/B978-0-12-819154-5.00023-0 (cit. on pp. 35–37, 39).

[46] J. E. van Engelen and H. H. Hoos. "A Survey on Semi-Supervised Learning". In: *Machine Learning* 109 (2020), pp. 373–440. DOI: 10.1007/s10994-019-05855-6 (cit. on p. 38).

[47] Saikat Dutt, Subramanian Chandramouli, and Amit Kumar Das. *Machine Learning*. Pearson Education India, 2018. ISBN: 978-93-5306-669-7 (cit. on p. 38).

[48] Paolo Dell'Aversana. "Reinforcement Learning in Optimization Problems. Applications to Geophysical Data Inversion". In: *AIMS Geosciences* 8.3 (2022), pp. 488–502 (cit. on p. 38).

[49] Sotiris B. Kotsiantis, Ioannis Zaharakis, and P. Pintelas. "Supervised Machine Learning: A Review of Classification Techniques". In: *Emerging Artificial Intelligence Applications in Computer Engineering* 160.1 (2007), pp. 3–24 (cit. on pp. 40, 43).

[50] Salvador García et al. "Big Data Preprocessing: Methods and Prospects". In: *Big Data Analytics* 1.1 (2016), p. 9. DOI: 10.1186/s41044-016-0014-0 (cit. on p. 42).

[51] C. K. Rodríguez. *A Computational Environment for Data Preprocessing in Supervised Classification*. Mayaguez, Puerto Rico: University of Puerto Rico, 2004 (cit. on p. 42).

[52] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Panagiotis E Pintelas. "Data Preprocessing for Supervised Learning". In: *International Journal of Computer Science* 1.2 (2006), pp. 111–117 (cit. on p. 42).

[53] Ana Nikolikj et al. "Assessing the Generalizability of a Performance Predictive Model". In: *arXiv preprint arXiv:2306.00040* (2023). DOI: 10.48550/arXiv.2306.00040. URL: https://arxiv.org/abs/2306.00040 (cit. on p. 43).

[54] Karthik Ramasubramanian and Abhishek Singh. "Machine Learning Model Evaluation". In: *Machine Learning Using R*. Berkeley, CA: Apress, 2017, pp. 425–464. ISBN: 978-1-4842-2334-5. DOI: 10.1007/978-1-4842-2334-5_7 (cit. on p. 43).

[55] Lior Rokach and Oded Maimon. "Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Springer, 2005, pp. 165–192. DOI: 10.1007/0-387-25465-X_9 (cit. on pp. 44–46).

[56] Muhammad Asfand Hafeez et al. "Performance improvement of decision tree: A robust classifier using tabu search algorithm". In: *Applied Sciences* 11.15 (2021), p. 6728. DOI: 10.3390/app11156728 (cit. on p. 45).

[57] Sotiris B. Kotsiantis. "Decision Trees: A Recent Overview". In: *Artificial Intelligence Review* 39 (2013), pp. 261–283. DOI: 10.1007/s10462-011-9272-4 (cit. on p. 45).

[58] Anuja Priyam et al. "Comparative Analysis of Decision Tree Classification Algorithms". In: *International Journal of Current Engineering and Technology* 3.2 (2013), pp. 334–337 (cit. on p. 46).

[59] Abdelhamid Djeffal. "Cours Fouille de Données Avancée". In: *Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie, Université Mohamed Khider-Biskra* (2014), pp. 6–8 (cit. on pp. 47, 51, 90).

[60] ListenData. *Support Vector Machine in R - Tutorial*. Accessed: 2025-07-14. 2017 (cit. on p. 48).

[61] Facundo Bre, Juan M. Gimenez, and Víctor D. Fachinotti. "Prediction of Wind Pressure Coefficients on Building Surfaces Using Artificial Neural Networks". In: *Energy and Buildings* 158 (2018), pp. 1429–1441. DOI: 10.1016/j.enbuild.2017.11.045 (cit. on p. 50).

[62] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18.7 (2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527 (cit. on p. 52).

[63] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003 (cit. on p. 53).

[64] Ayilobeni Kikon and Paresh Chandra Deka. "Artificial intelligence application in drought assessment, monitoring and forecasting: a review". In: *Stochastic Environmental Research and Risk Assessment* 36.5 (2022), pp. 1197–1214. DOI: 10.1007/s00477-021-02129-3 (cit. on p. 54).

[65] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539 (cit. on p. 55).

[66] Abhishek Raj. *Convolutional Neural Networks (CNN) Architectures Explained*. https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243. Accessed: July 14, 2025. 2023 (cit. on p. 57).

[67] Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal. "Machine Learning for Routing in IoT: A Review". In: *Recent Advances in Communication Technology, Computing and Engineering*. Ed. by Mariyam Ouaissa et al. RGN Publications, 2021, pp. 195–207. ISBN: 978-81-954166-0-8. DOI: 10.26713/978-81-954166-0-8 (cit. on p. 58).

[68] Nour El Houda Larouci, Somia Sahraoui, and Abdelhamid Djeffal. "A Survey of Deep Learning Solutions for Network Routing in Modern Networks: Challenges and Opportunities". In: *2025 International Symposium on iNnovative Informatics of Biskra (ISNIB)*. IEEE. 2025, pp. 1–6. DOI: 10.1109/ISNIB64820.2025.10983118 (cit. on p. 58).

[69] J. Boyan and M. Littman. "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach". In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Available online. Morgan-Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper_files/paper/1993/file/4ea06fbc83cdd0a06020c35d50e1e89a-Paper.pdf (cit. on p. 58).

[70] Z. Mammeri. "Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches". In: *IEEE Access* 7 (2019), pp. 55916–55950. DOI: 10.1109/ACCESS.2019.2913776 (cit. on p. 59).

[71] S. Choi and D.-Y. Yeung. "Predictive Q-Routing: A Memory-Based Reinforcement Learning Approach to Adaptive Traffic Control". In: *Advances in Neural Information Processing Systems*. Vol. 8. 1995 (cit. on p. 59).

[72] S. Kumar and R. Miikkulainen. "Dual Reinforcement Q-Routing: An On-line Adaptive Routing Algorithm". In: *Proceedings of the Artificial Neural Networks in Engineering Conference*. 1997, pp. 231–238 (cit. on p. 59).

[73] R. Sun, S. Tatsumi, and G. Zhao. "Application of Multiagent Reinforcement Learning to Multicast Routing in Wireless Ad Hoc Networks Ensuring Resource Reservation". In: *IEEE International Conference on Systems, Man and Cybernetics*. Vol. 1. 2002, pp. 409–414. DOI: 10.1109/ICSMC.2002.1168009 (cit. on p. 59).

[74] Y.-H. Chang, T. Ho, and L. P. Kaelbling. "Mobilized Ad-Hoc Networks: A Reinforcement Learning Approach". In: *Proceedings of the International Conference on Autonomic Computing*. 2004, pp. 240–247. DOI: 10.1109/ICAC.2004.1301369 (cit. on p. 59).

[75] D. Chetret, C.-K. Tham, and L. W.-C. Wong. "Reinforcement Learning and CMAC-Based Adaptive Routing for MANETs". In: *Proceedings of the 2004 12th IEEE International Conference on Networks (ICON 2004)*. Vol. 2. 2004, pp. 540–544. DOI: 10.1109/ICON.2004.1409226 (cit. on p. 59).

[76] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. *RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing*. 2003 (cit. on pp. 59, 84).

[77] Y. Zhang and M. Fromherz. "Constrained Flooding: A Robust and Efficient Routing Framework for Wireless Sensor Networks". In: *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*. Vol. 1. 2006, 6–pp. DOI: 10.1109/AINA.2006.132 (cit. on p. 59).

[78] P. Wang and T. Wang. "Adaptive Routing for Sensor Networks Using Reinforcement Learning". In: *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT 2006)*. 2006, p. 219. DOI: 10.1109/CIT.2006.34 (cit. on p. 59).

[79] Shaoqiang Dong, Prathima Agrawal, and Krishna Sivalingam. "Reinforcement Learning Based Geographic Routing Protocol for UWB Wireless Sensor Network". In: *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*. 2007, pp. 652–656. DOI: 10.1109/GLOCOM.2007.127 (cit. on p. 59).

[80] X. Liang, I. Balasingham, and S. Byun. "A Reinforcement Learning Based Routing Protocol with QoS Support for Biomedical Sensor Networks". In: *Proceedings of the 1st International Symposium on Applied Sciences on Biomedical and Communication Technologies*. 2008, pp. 1–5. DOI: 10.1109/ISABEL.2008.4712578 (cit. on p. 59).

[81] P. Fu, J. Li, and D. Zhang. "Heuristic and Distributed QoS Route Discovery for Mobile Ad Hoc Networks". In: *Proceedings of the 5th International Conference on Computer and Information Technology (CIT'05)*. 2005, pp. 512–516. DOI: 10.1109/CIT.2005.125 (cit. on p. 59).

[82] K. Wang, W. Wong, and T. Y. Chai. "A MANET Routing Protocol Using Q-Learning Method Integrated with Bayesian Network". In: *Proceedings of the 2012 IEEE International Conference on Communication Systems (ICCS)*. 2012, pp. 270–274. DOI: 10.1109/ICCS.2012.6406152 (cit. on p. 59).

[83] Sudhir K. Routray and K. P. Sharmila. "Routing in Dynamically Changing Node Location Scenarios: A Reinforcement Learning Approach". In: *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*. IEEE, 2017, pp. 458–462. DOI: 10.1109/AEEICB.2017.7972354 (cit. on p. 59).

[84] Tiansi Hu and Yunsi Fei. "An Adaptive and Energy-Efficient Routing Protocol Based on Machine Learning for Underwater Delay Tolerant Networks". In: *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecom-*

munication Systems. 2010, pp. 381–384. DOI: 10.1109/MASCOTS.2010.45 (cit. on p. 60).

[85] Brian Russell, Michael L. Littman, and Wade Trappe. "Integrating Machine Learning in Ad Hoc Routing: A Wireless Adaptive Routing Protocol". In: *International Journal of Communication Systems* 24.7 (2011), pp. 950–966. DOI: 10.1002/dac.1202 (cit. on p. 60).

[86] Chetana V. Murudkar and Richard D. Gitlin. "Optimal-Capacity, Shortest Path Routing in Self-Organizing 5G Networks Using Machine Learning". In: *2019 IEEE 20th Wireless and Microwave Technology Conference (WAMICON)*. IEEE, 2019, pp. 1–5. DOI: 10.1109/WAMICON.2019.8765434 (cit. on p. 60).

[87] O. S. Gnana Prakasi and P. Varalakshmi. "Decision Tree Based Routing Protocol (DTRP) for Reliable Path in MANET". In: *Wireless Personal Communications* 109.1 (Nov. 2019), pp. 257–270. ISSN: 1572-834X. DOI: 10.1007/s11277-019-06563-z (cit. on p. 60).

[88] Farooque Hassan Kumbhar and Soo Young Shin. "DT-VAR: Decision Tree Predicted Compatibility-Based Vehicular Ad-Hoc Reliable Routing". In: *IEEE Wireless Communications Letters* 10.1 (2021), pp. 87–91. DOI: 10.1109/LWC.2020.3021430 (cit. on p. 60).

[89] Pawan Kumar Sharma, S. C. Mahajan, and Reena Jain. "Alternate Path Finding System Using Support Vector Classifier in Mobile Ad Hoc Network". In: *International Journal of Application or Innovation in Engineering & Management (IJAIEM)* 3.6 (June 2014), pp. 87–94. URL: http://www.ijaiem.org/ (cit. on p. 61).

[90] Feeza Khan, Saira Memon, and Sana Hoor Jokhio. "Support Vector Machine Based Energy Aware Routing in Wireless Sensor Networks". In: *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*. IEEE, 2016, pp. 1–4. DOI: 10.1109/ICRAI.2016.7791218 (cit. on p. 61).

[91] Liang Zhao et al. "A SVM Based Routing Scheme in VANETs". In: *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*. IEEE, 2016, pp. 380–383. DOI: 10.1109/ISCIT.2016.7751655 (cit. on p. 61).

[92] Praveena Akki and V. Vijayarajan. "Machine Learning Algorithm-Based Minimisation of Network Traffic in Mobile Cloud Computing". In: *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology (ICDECT*

2017). Springer, 2019, pp. 573–584. DOI: 10.1007/978-981-13-1610-4_58 (cit. on p. 61).

[93] Mohamed Mira et al. "Optimization of IoT Routing Based on Machine Learning Techniques: Case Study of Passenger Flow Control in Airport 3.0". In: *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*. IEEE, 2018, pp. 136–141. DOI: 10.1109/IINTEC.2018.8695294 (cit. on p. 61).

[94] Leticia Lemus Cárdenas, Juan Pablo Astudillo León, and Ahmad Mohamad Mezher. "GraTree: A Gradient Boosting Decision Tree Based Multimetric Routing Protocol for Vehicular Ad Hoc Networks". In: *Ad Hoc Networks* 137 (2022), p. 102995. DOI: 10.1016/j.adhoc.2022.102995 (cit. on p. 61).

[95] Vidushi Vashishth, Anshuman Chhabra, and Deepak Kumar Sharma. "A Machine Learning Approach Using Classifier Cascades for Optimal Routing in Opportunistic Internet of Things Networks". In: *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2019, pp. 1–9. DOI: 10.1109/SAHCN.2019.8824952 (cit. on p. 62).

[96] O. S. Eyobu and K. Edwinah. "A Deep Learning-Based Routing Approach for Wireless Mesh Backbone Networks". In: *IEEE Access* 11 (2023), pp. 49509–49518 (cit. on p. 62).

[97] M. Nagalingayya and B. S. Mathpati. "Energy-Efficient Cooperative Routing Scheme with Recurrent Neural Network Based Decision Making System for Wireless Multimedia Sensor Networks". In: *Multimedia Tools and Applications* 81.27 (2022), pp. 39785–39801. DOI: 10.1007/s11042-022-12938-5 (cit. on p. 62).

[98] A. C. Sumathi et al. "NEWTR: A Multipath Routing for Next Hop Destination in Internet of Things with Artificial Recurrent Neural Network (RNN)". In: *International Journal of Machine Learning and Cybernetics* 13.10 (2022), pp. 2869–2889. DOI: 10.1007/s13042-022-01568-w (cit. on p. 62).

[99] T. M. Modi and P. Swain. "Enhanced Routing Using Recurrent Neural Networks in Software Defined-Data Center Network". In: *Concurrency and Computation: Practice and Experience* 35.5 (2023), e7557. DOI: 10.1002/cpe.7557 (cit. on p. 62).

[100] A. Pal et al. "A Multipath Load Balancing Routing Protocol in Mobile Ad Hoc Network Using Recurrent Neural Network". In: *Computational Intelligence, Communications, and Business Analytics: Second International Conference, CICBA 2018, Kalyani, India, July 27–28, 2018, Revised Selected Papers, Part I*. 2019, pp. 458–464. DOI: 10.1007/978-981-13-8578-0_36 (cit. on p. 62).

[101] A. Venkatesh and S. Asha. "DERNNet: Dual Encoding Recurrent Neural Network Based Secure Optimal Routing in WSN". In: *Computer Systems Science and Engineering* 45.2 (2023), pp. 1375–1392 (cit. on p. 62).

[102] T. M. Modi and P. Swain. "Intelligent Routing Using Convolutional Neural Network in Software-Defined Data Center Network". In: *The Journal of Supercomputing* 78 (2022), pp. 13373–13392. DOI: 10.1007/s11227-022-04348-z (cit. on p. 63).

[103] T. D. H. Hussein et al. "BA-CNN: Bat Algorithm-Based Convolutional Neural Network Algorithm for Ambulance Vehicle Routing in Smart Cities". In: *Mobile Information Systems* 2022.1 (2022), p. 7339647. DOI: 10.1155/2022/7339647 (cit. on p. 63).

[104] F. Geyer and G. Carle. "Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning". In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. 2018, pp. 40–45 (cit. on p. 63).

[105] B. Yan et al. "Flowlet-Level Multipath Routing Based on Graph Neural Network in OpenFlow-Based SDN". In: *Future Generation Computer Systems* 134 (2022), pp. 140–153. DOI: 10.1016/j.future.2022.04.006 (cit. on p. 63).

[106] H. Wei, Y. Zhao, and K. Xu. "G-Routing: Graph Neural Networks-Based Flexible Online Routing". In: *IEEE Network* 37.4 (2023), pp. 90–96. DOI: 10.1109/MNET.012.2300052 (cit. on p. 63).

[107] K. Rusek et al. "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN". In: *IEEE Journal on Selected Areas in Communications* 38.10 (2020), pp. 2260–2270. DOI: 10.1109/JSAC.2020.3000405 (cit. on p. 63).

[108] D. Liu et al. "Deep-Learning-Aided Packet Routing in Aeronautical Ad Hoc Networks Relying on Real Flight Data: From Single-Objective to Near-Pareto Multiobjective Optimization". In: *IEEE Internet of Things Journal* 9.6 (2021), pp. 4598–4614 (cit. on p. 63).

[109] J. Reis et al. "Deep Neural Networks for Network Routing". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–8 (cit. on p. 64).

[110] A. Swaminathan et al. "GraphNET: Graph Neural Networks for Routing Optimization in Software Defined Networks". In: *Computer Communications* 178 (2021), pp. 169–182. DOI: 10.1016/j.comcom.2021.07.025 (cit. on p. 64).

[111] B. Chen et al. "An Approach to Combine the Power of Deep Reinforcement Learning with a Graph Neural Network for Routing Optimization". In: *Electronics* 11.3 (2022), p. 368 (cit. on p. 64).

[112] O. Hope and E. Yoneki. "GDDR: GNN-Based Data-Driven Routing". In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 2021, pp. 517–527. DOI: 10.1109/ICDCS51616.2021.00056 (cit. on p. 64).

[113] M. Ding et al. "GROM: A Generalized Routing Optimization Method With Graph Neural Network and Deep Reinforcement Learning". In: *Journal of Network and Computer Applications* 229 (2024), p. 103927. DOI: 10.1016/j.jnca.2024.103927 (cit. on p. 64).

[114] P. Almasan et al. "Deep Reinforcement Learning Meets Graph Neural Networks: Exploring a Routing Optimization Use Case". In: *Computer Communications* 196 (2022), pp. 184–194 (cit. on p. 65).

[115] P. Swain et al. "CoDRL: Intelligent Packet Routing in SDN Using Convolutional Deep Reinforcement Learning". In: *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. 2019, pp. 1–6 (cit. on p. 65).

[116] J. Chafla Altamirano et al. "Routing Optimization Based on DRL and Generative Adversarial Networks for SDN Environments". In: *NOMS 2024–2024 IEEE Network Operations and Management Symposium*. 2024, pp. 1–5 (cit. on p. 65).

[117] X. You et al. "Toward Packet Routing With Fully Distributed Multiagent Deep Reinforcement Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.2 (2020), pp. 855–868 (cit. on p. 65).

[118] Z. Rao, Y. Xu, and S. Pan. "A Deep Learning-Based Constrained Intelligent Routing Method". In: *Peer-to-Peer Networking and Applications* 14.4 (2021), pp. 2224–2235 (cit. on p. 65).

[119] *The Network Simulator - ns-2*. Available at https://www.isi.edu/nsnam/ns/. Accessed on: June 6, 2022. (cit. on pp. 83, 109).

[120] *Machine Learning at Waikato University*. Available at https://www.cs.waikato.ac.nz/ml/index.html. Accessed on: June 6, 2022. (cit. on p. 89).

[121] Leticia Lemus Cárdenas, Juan Pablo Astudillo León, and Ahmad Mohamad Mezher. "GraTree: A Gradient Boosting Decision Tree Based Multimetric Routing Protocol for Vehicular Ad Hoc Networks". In: *Ad Hoc Networks* 137 (2022), p. 102995 (cit. on p. 95).

[122] Prathapchandran Kannimuthu and Janani Thangamuthu. "Decision Tree Trust (DTTrust)-Based Authentication Mechanism to Secure RPL Routing Protocol on Internet of Battlefield Thing (IoBT)". In: *International Journal of Business Data Communications and Networking (IJBDCN)* 17.1 (2021), pp. 1–23 (cit. on p. 95).