UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Klea Elmazi

A Federated Learning Framework for Explainable Proactive Task Offloading in IoT Systems

DOCTORAL THESIS

Supervisor:
Prof. dr. sc. Jonatan Lerga

Rijeka, 2026

Doctoral thesis supervisor: Prof. Jonatan Lerga, PhD (Faculty of Engineering, University of Rijeka)

The doctoral thesis was defended on _____ at the University of Rijeka, Faculty of Engineering, Croatia, in front of the following Evaluation Committee:

1. Prof.
2. Prof.
3. Prof.

Rijeka, 2026

Acknowledgements

This PhD journey has been one of the hardest and most meaningful things I have ever done, and I could not have done it alone. I want to start by thanking the Faculty of Engineering, University of Rijeka, for giving me the opportunity to carry out this research. Being part of this institution and its academic community has been a true privilege.

My deepest thanks go to my supervisor, Prof. Jonatan Lerga, who guided me through every stage of this work. He was patient when I was stuck, honest when I needed direction. He didn't just help me finish a PhD, he helped me grow as a researcher. I also want to thank all the professors and collegues who have been part of my academic path over the years. Every course I took, every conversation, every piece of feedback shaped the way I think and work today. I am thankful to each one of them.

To my parents and my brother, thank you for everything you have given me. You worked hard so I could have the chances you didn't, and you raised me to believe that I could do anything if I tried hard enough. This achievement is yours as much as it is mine.

To my parents-in-law, thank you for your constant support and kindness. Your encouragement throughout this journey meant more than you know.

To my husband Donald, thank you for standing by me through all of it. The stress, the late nights, the moments of doubt, the success, you were always there. You never let me feel like I was doing this alone. You inspire me to be the best version of myself. I am so grateful for you.

And to my daughter Hana, you are the reason I keep going. There are hard days when I want to stop, but then I think of you I find the strength to move forward. I

hope that one day, you are proud of me just as I am proud of you. Everything I do, I do it for you.

To everyone who gave me an opportunity that, in one way or another, led me to this moment, thank you.

Abstract

As Internet of Things (IoT) infrastructures and distributed intelligent systems continue to grow, there is an increasing need for efficient task offloading that can meet strict requirements in terms of latency, energy efficiency, and scalability. Centralized machine learning approaches are often limited by communication overhead, privacy concerns, and reduced adaptability to dynamic environments. This thesis addresses these challenges by proposing a Explainable Digital Twin-Federated Multi-Agent Reinforcement Learning (XDT-FMARL) framework for proactive task offloading in heterogeneous IoT networks.

The proposed methodology integrates (i) Federated Learning (FL) to enable scalable multi-device training without centralized data collection, (ii) Multi-Agent Reinforcement Learning (MARL) to optimize distributed offloading decisions, (iii) Digital Twin (DT) to provide predictive awareness of device and network states (e.g., battery evolution, CPU load, and latency), (iv) explainability mechanisms based on gradient-driven feature attribution to improve interpretability and trust, and (v) Large Language Model (LLM)–assisted context reasoning to support transparent, context-aware decision intelligence.

A comprehensive simulation environment is developed to model Non-Independent and Identically Distributed (non-IID) data, device behavior, fluctuating wireless conditions, and diverse workload profiles. Experimental evaluation compares the proposed approach against local-only and naïve offloading baselines across stable and unstable scenarios. Results demonstrate improved offloading stability and convergence through DT integration, enhanced scalability via federated policy aggregation, and increased transparency through explainability outputs that align with relevant system features. The findings validate the framework as a practical step toward

trustworthy and proactive offloading in DT–driven, privacy-aware IoT ecosystems.

Keywords: Internet of Things (IoT), Distributed Intelligence, Computation Offloading, Federated Learning, Digital Twins, Multi-Agent Reinforcement Learning, Large Language Models, Edge Computing.

Sažetak

Tijekom posljednjeg desetljeća, Internet stvari (IoT) razvio se u opsežan distribuirani računalni ekosustav koji povezuje senzore, aktuatore i ugradbenu inteligenciju u različitim domenama, uključujući industrijsku automatizaciju, pametne gradove, prometne sustave i inteligentnu infrastrukturu. Ovi se sustavi oslanjaju na uređaje ograničenih resursa koji kontinuirano generiraju velike količine podataka, istovremeno izvršavajući zadatke očitanja, obrade i upravljanja uz stroga ograničenja latencije i potrošnje energije. S porastom broja povezanih uređaja, učinkovito upravljanje računalnim i komunikacijskim resursima postaje ključan izazov. Posebno je važno odrediti kada se zadaci trebaju izvršavati lokalno, a kada ih treba premjestiti na rubne ili resurse u oblaku, kako bi se očuvao odziv sustava i produžio vijek trajanja baterije uređaja.

Tradicionalne centralizirane računalne arhitekture sve su manje prikladne za suvremene IoT sustave. Slanje svih podataka na udaljene poslužitelje u oblaku uzrokuje povećanu latenciju, zagušenje mreže i visoke komunikacijske troškove. S druge strane, izvršavanje svih zadataka lokalno na uređajima brzo iscrpljuje baterijske resurse i ograničava složenost modela strojnog učenja. Zbog toga su se rubno računarstvo (engl. edge computing) i federirano učenje (engl. federate learning - FL) nametnuli kao ključni pristupi za distribuiranu inteligenciju. Rubno računarstvo omogućuje obradu podataka bliže njihovom izvoru, dok FL omogućuje zajedničko treniranje modela bez centralizacije sirovih podataka. Međutim, stvarni IoT sustavi karakterizirani su visokom heterogenošću i dinamikom: uređaji imaju različite hardverske mogućnosti, bežične veze su promjenjive, opterećenja variraju, a podaci su često neovisni i nejednako distribuirani (non-IID). Ovi čimbenici značajno otežavaju koordinaciju učenja i raspoređivanje zadataka.

Ova disertacija adresira navedene izazove predlažući integrirani okvir za proaktivno raspoređivanje zadataka u distribuiranim IoT sustavima. Predloženi okvir kombinira prediktivne digitalne blizance (engl. digital twin - DT), federirano višeagentno pojačano učenje (FMARL), mehanizme objašnjivog odlučivanja te zaključivanje uz pomoć velikih jezičnih modela (engl. large language model - LLM). Glavni cilj istraživanja je omogućiti IoT uređajima donošenje prilagodljivih, adaptivnih odluka o raspoređivanju zadataka tako da optimiziraju potrošnju energije, računalno opterećenje i komunikacijsku latenciju, uz očuvanje transparentnosti procesa odlučivanja.

U središtu predložene arhitekture nalazi se digitalni blizanac svakog IoT uređaja koji odražava njegovo operativno stanje. Digitalni blizanac kontinuirano prati parametre sustava poput razine baterije, iskorištenosti procesora, veličine zadataka i mrežne latencije. Primjenom regresijskih modela predviđaju se kratkoročni trendovi dostupnosti resursa, čime se omogućuje anticipacija budućih stanja poput pražnjenja baterije, preopterećenja procesora ili zagušenja mreže. Time se odluke o raspoređivanju zadataka donose proaktivno, a ne reaktivno, što doprinosi stabilnosti sustava i učinkovitijem korištenju resursa.

Na temelju tih prediktivnih mogućnosti, predloženi okvir koristi FMARL za koordinaciju odluka među uređajima. Svaki uređaj djeluje kao autonomni agent koji uči kada izvršiti zadatak lokalno, a kada ga proslijediti na rubne poslužitelje. Umjesto centraliziranog učenja, agenti surađuju putem federiranog mehanizma razmjenom modelskih parametara, a ne sirovih podataka. Ovakav pristup poboljšava skalabilnost i očuva privatnost podataka, omogućujući konvergenciju prema zajedničkoj politici u heterogenim uvjetima.

Radi osiguravanja transparentnosti i pouzdanosti, u predloženi okvir su integrirane metode objašnjive umjetne inteligencije. Analiza salijentnosti temeljena na gradijentima koristi se za kvantifikaciju utjecaja ulaznih značajki na odluke o raspoređivanju zadataka. Time se omogućuje uvid u doprinos varijabli poput razine baterije, opterećenja procesora, mrežne latencije i dinamike opterećenja. Ovi rezultati omogućuju provjeru usklađenosti odluka s inženjerskim zahtjevima, primjerice očuvanjem energije pri niskoj razini baterije ili izbjegavanjem prijenosa podataka

tijekom mrežnog zagušenja.

Iako metode pojačanog učenja nude snažne optimizacijske mogućnosti, njihovo treniranje u dinamičkim okruženjima može biti računalno zahtjevno i osjetljivo na hiperparametre. Kako bi se to ublažilo, predloženi okvir uključuje dodatni sloj odlučivanja temeljen na LLM-ima. Strukturirani upiti generiraju se na temelju informacija iz digitalnih blizanaca, stanja uređaja i federiranih metrika. LLM koristi kontekstualno zaključivanje za donošenje odluka bez potrebe za ponovnim treniranjem modela. Ovakva hibridna arhitektura kombinira adaptivnost pojačanog učenja i fleksibilnost jezičnih modela.

Za evaluaciju predloženog okvira razvijeno je simulacijsko okruženje koje modelira heterogene IoT uređaje, stohastičku mrežnu latenciju, varijabilna opterećenja i dinamiku potrošnje energije. Eksperimenti su provedeni u različitim scenarijima, uključujući stabilne i nestabilne mrežne uvjete, različita opterećenja i konfiguracije uređaja.

Rezultati pokazuju da predloženi okvir postiže uravnotežen kompromis između smanjenja latencije i energetske učinkovitosti. U standardnim mrežnim uvjetima, LLM-potpomognuti kontroler postiže prosječnu latenciju od približno 252 ms uz potrošnju energije od oko 0,125 J po odluci. Lokalno izvršavanje rezultira većom latencijom (oko 441 ms), dok pristup s potpunim rasterešenjem povećava potrošnju energije (oko 0,153 J). U svim scenarijima predloženi sustav održava razinu baterije iznad 80%, prilagođavajući odluke dinamičkim uvjetima.

Zaključno, rezultati ove disertacije potvrđuju učinkovitost integracije prediktivnog modeliranja, distribuiranog učenja i objašnjivog odlučivanja u jedinstveni okvir za upravljanje resursima u IoT sustavima. Predloženi pristup pokazuje da strategije za raspoređivanje zadataka koje su proaktivne, transparentne i usmjerene na očuvanje privatnosti mogu značajno poboljšati učinkovitost i pouzdanost distribuiranih sustava te predstavljaju temelj za daljnja istraživanja u području skalabilne i pouzdane inteligencije na rubu mreže. (Ovdje umetnite hrvatski sažetak.)

Contents

List of Figures

List of Tables

List of Abbreviations

ARIMA Autoregressive Integrated Moving Average

CPU Central Processing Unit

D3QN Dueling Double Deep Q-Network

DDQN Double Deep Q-Network

DQN Deep Q-Network

DT Digital Twin

FedAvg Federated Averaging

FL Federated Learning

IoT Internet of Things

LLM Large Language Model

MARL Multi-Agent Reinforcement Learning

MDP Markov Decision Process

ML Machine Learning

non-IID non-Independent and Identically Distributed

PER Prioritized Experience Replay

QoS Quality of Service

RL Reinforcement Learning

SMA Simple Moving Average

XAI Explainable Artificial Intelligence

XDT-FMARL Explainable Digital Twin-Federated Multi-Agent Reinforcement Learning

Chapter 1

Introduction

1.1 Context and Motivation

The IoT has evolved from an abstract idea to a widely deployed technological backbone that links sensors, actuators, and embedded intelligence across domains such as residential environments, industrial systems, transportation networks, healthcare services, and urban infrastructures [1]. Contemporary IoT ecosystems [2] are made up of vast collections of diverse devices limited in resources that continuously emit high-rate data streams while simultaneously needing to execute sensing, learning, decision-making, and control operations with strict real-time requirements [3]. As these deployments scale in size and complexity, the computational workload, communication traffic, and energy consumption demanded of each participating device correspondingly increase, intensifying the pressure on their limited resources [4].

Traditional centralized data processing architectures [5, 6] are less appropriate for these circumstances. Sending all collected data to remote cloud servers introduces substantial latency, aggravates network congestion, and results in prohibitive energy usage. In contrast, performing all computation directly on end devices rapidly depletes their batteries and restricts the sophistication and scale of the models that can be executed. In response, edge computing [7, 8]and FL have gained prominence as foundational technologies for implementing scalable intelligence in IoT ecosystems [9]. By distributing data processing and model training across IoT devices and nearby edge servers, FL enables collaborative learning without sharing raw data,

thus improving privacy and reducing communication overhead. Despite these advantages, the implementation of FL in real-world IoT scenarios remains challenging, mainly due to the heterogeneity of the device and hardware, fluctuating and unreliable network conditions, and stringent constraints on computation, memory, and energy resources [10].

At the same time, modern IoT deployments are making greater use of virtual sensing, where physical sensors are augmented with virtual counterparts that estimate missing, unreliable, or hidden measurements based on existing data streams. These virtual sensors increase data coverage, enhance overall system resilience, and produce more comprehensive situational awareness [11]. Nevertheless, they also introduce higher demands on computation and generate more intensive communication traffic across the system. As a result, there is a growing need for smart and efficient task offloading mechanisms that can dynamically decide whether computations should run on local devices, be offloaded to nearby edge servers, or be postponed for later execution [12] [13].

DTs offer a powerful conceptual tool for solving this problem. A DT is a virtual counterpart of a physical device or system that is continuously updated to represent its current operating conditions, including metrics such as battery status, CPU utilization, and network quality [14]. In addition to simply tracking these real-time states, DTs support predictive analytics by running lightweight models, such as linear regression or moving-average-based predictors, to estimate short-term trends in energy usage, communication latency, and workload fluctuations [15].

When integrated with edge computing infrastructures, DTs create a "mirror world" in which potential decisions and control policies can be tested, evaluated, and optimized before being applied to the real system. This ability to conduct virtual trials reduces operational risk and improves the reliability and effectiveness of decision-making [16]. Earlier research has shown that embedding DTs within Reinforcement Learning (RL)–driven computation offloading frameworks can significantly enhance the decision-making process and overall system performance. In particular, improvements in cumulative reward exceeding 38% have been reported compared to methods that rely exclusively on unprocessed direct observations from

the physical environment, underscoring the value of simulation driven by DT in the guide of learning and control [17].

This thesis introduces a FL framework for explainable proactive task offloading in IoT systems that integrates FL, DT, and RL to optimize energy consumption, latency, and learning performance in IoT systems. The proposed offloading strategies are shown to maintain device battery levels above 85% while significantly reducing CPU utilization compared to naïve offloading and purely local processing. The inclusion of Explainable AI (XAI) [18, 19] enables transparent interpretation of offloading decisions, thereby enhancing trust and accountability in autonomous system behavior [20]. However, RL–based controllers suffer from several limitations [21]. They require extensive training, careful hyperparameter tuning, and large amounts of interaction data, and they adapt slowly in highly non-stationary environments. When network conditions, device populations, or workload patterns change, policies often need to be retrained or re-federated, introducing significant overhead.

Recent progress in LLMs[22] introduces a fundamentally new paradigm. LLMs possess strong in-context learning abilities: when provided with carefully structured prompts that encode system states, examples, and operational constraints, they can deduce appropriate decisions and actions without any gradient-based fine-tuning or retraining. An increasing number of studies indicate that, in DT-enabled edge networks, LLMs can act as drop-in replacements for traditional MARL controllers [23, 24] by embedding DT states [25, 51], historical actions, and feedback into the prompt [26]. In doing so, they can achieve performance on par with, or even superior to, well-trained RL agents [27] while avoiding lengthy and computation-heavy training procedures. This development opens a distinctive opportunity to tightly integrate DT, FL, and LLMs to realize offloading strategies in IoT environments that are not only adaptive to changing conditions, but also energy-efficient and inherently more interpretable and explainable to system designers and operators.

1.2 Research Aims, Hypotheses, and Scientific Contributions

The main objective of this doctoral thesis is to develop and evaluate an integrated framework, termed XDT-FMARL, for proactive and energy-aware task offloading in IoT networks. This framework is intended to tackle the core problem of achieving intelligent, transparent and adaptive decision making in large-scale, heterogeneous, and resource-limited IoT settings, where devices must continuously and autonomously manage trade-offs among computational workload, energy reserves, and dynamic network conditions to maintain efficient and reliable operation.

More precisely, the research seeks to establish a unified framework in which DTs offer predictive visibility into near-term system behavior, FL enables joint model training across geographically dispersed devices without requiring raw data aggregation, and MARL drives flexible, decentralized offloading strategies that adapt to changing conditions. These core elements are complemented by XAI techniques [28], which ensure that the decisions produced by the system are not only high-performing but also understandable and open to inspection. By embedding explainability into each step of the decision-making pipeline, the proposed framework seeks to build stronger user and stakeholder trust, enhance transparency and accountability, and ultimately support more reliable, effective deployment and day-to-day operation of IoT systems in realistic, complex, and large-scale environments.

Within this context, the research focuses on enabling proactive task offloading, whereby decisions are informed not only by current system states but also by predicted future conditions such as battery depletion, CPU load, and network latency. Through the integration of DT forecasts into the learning and decision-making process, the framework aims to anticipate resource constraints and prevent performance degradation before it occurs. At the same time, FL allows multiple IoT devices to collectively improve their offloading policies while preserving decentralized operation.

Finally, this thesis seeks to show that the proposed XDT-FMARL framework can deliver a more favorable trade-off among energy consumption, computational

throughput, and interpretability of decisions than traditional offloading and learning solutions. By tightly integrating predictive modeling, distributed learning, adaptive control, and explainable decision mechanisms within a unified architecture, this research aspires to define a new benchmark for intelligent, reliable, and environmentally sustainable computation management in emerging and large-scale IoT ecosystems, paving the way for more accountable and efficient next-generation network infrastructures.

Research Hypotheses The research hypotheses of this dissertation are formulated around the core components of the proposed framework. These hypotheses formalize the expected impact of FL, DTs, and XAI on the effectiveness, stability, and trustworthiness of task offloading in IoT networks.

- **Hypothesis 1 – Digital Twin-Enhanced Learning.**
  Integrating DT forecasts into MARL training will improve stability and accuracy of offloading decisions under fluctuating latency, CPU variations, and battery conditions.

- **Hypothesis 2 – Federated MARL & Privacy.**
  Federated training of distributed MARL agents will maintain global policy convergence without sharing raw sensor data, thereby improving privacy.

- **Hypothesis 3 – Explainability & Trust.**
  Using gradient-based explainability methods will increase the interpretability and trust of offloading decisions, particularly in high-demand scenarios.

By providing forward-looking estimates of key system metrics, such as future battery states, processor load, and communication latency, the DT allows agents to proactively account for upcoming resource variations instead of simply responding to changes after they occur. This predictive capability supports the derivation of offloading strategies that remain reliable, resource-aware, and performance-efficient even when operating conditions fluctuate rapidly and unpredictably over time.

Federated training of distributed MARL agents is expected to drive convergence toward a unified global offloading policy while still allowing each agent to operate in a decentralized manner. Instead of transmitting raw sensor measurements, devices share locally computed model updates, enabling the federated framework to retain learning effectiveness across diverse and heterogeneous IoT nodes. This strategy mitigates communication overhead, scalability bottlenecks, and privacy concerns associated with fully centralized training. Consequently, the working assumption is that federated MARL can deliver performance comparable to, or even better than, centralized learning when applied to large-scale, resource-limited network environments.

The integration of gradient-based explainability techniques is anticipated to enhance both the interpretability and perceived reliability of the offloading choices generated by the framework. By quantifying how strongly key system parameters, such as battery state, processor utilization, and communication latency contribute to each individual decision, these explainability tools offer clear and systematic visibility into the agent's internal reasoning. This becomes particularly important in settings that demand substantial resources or involve significant safety risks, where stakeholders need to examine how autonomous decisions are made, explain and defend those decisions, and ultimately place justified trust in the system's behavior.

Together, these hypotheses define the scientific foundation of this dissertation and serve as the basis for the experimental design and evaluation presented in subsequent chapters.

Scientific Contributions This doctoral thesis makes several original scientific contributions to the fields of IoT, distributed artificial intelligence, and adaptive edge computing. The contributions are listed as follows:

- An architecture for proactive task offloading in IoT systems that integrates Federated MARL, Digital Twins, and LLM-guided decision support.
- A method for explainable IoT task offloading based on gradient-based saliency mapping.

- An LLM-guided decision support method for proactive task offloading in IoT systems.

The primary contribution lies in designing a unified and transparent framework for proactive task offloading in IoT environments that jointly leverages DT, Federated MARL and LLM–driven decision support. In contrast to previous work that typically treats these technologies as separate building blocks, the proposed framework defines an integrated architecture where predictive modeling, decentralized learning, and context-aware decision-making are tightly interlinked and continuously inform each other. By maintaining this tight integration, IoT systems can anticipate and predict future resource constraints, coordinate collaborative learning among widely distributed devices, and dynamically plan, prioritize, and offload computation. In doing so, they can minimize energy consumption while still supporting large-scale, diverse, and rapidly evolving deployments.

A second key contribution is the development of a new explainability framework for IoT task offloading that leverages gradient-based saliency analysis. This framework systematically quantifies how critical system parameters, such as battery level, processor load, and communication delay shape offload decisions. By taking this approach, learning-based controllers that would otherwise function as opaque black boxes are transformed into decision-making mechanisms that remain understandable to humans and can be systematically examined, verified and validated using formal methods. In addition to identifying which variables are most influential, the method provides detailed, instance-level attribution of each decision, allowing users and operators to see precisely which input conditions had the greatest impact on a particular offloading outcome. This form of interpretability offers a clear, traceable link from inputs to decisions. Such transparency is especially critical in distributed, heterogeneous IoT environments, where device capabilities can change dynamically, network performance is highly variable, and computational as well as energy resources are tightly constrained, making informed and trustworthy offloading decisions essential.

Overall, this work advances the domain of explainable distributed intelligence by introducing a systematic framework for transparent, data-driven reasoning in com-

plex, fast-evolving environments. In these settings, the ability to interpret, clarify, and justify autonomous decisions is essential for building and maintaining user trust, safeguarding operational safety, supporting efficient debugging and formal system auditing, and, ultimately, making it feasible to deploy robust, reliable solutions at scale in real-world conditions.

The third main contribution is the design of an LLM-driven decision support framework for proactive task offloading in federated IoT environments. In this paradigm, real-time insights from DT models and metrics from FL processes are translated into structured natural-language prompts. These prompts are then provided to an LLM, which uses in-context reasoning to derive appropriate offloading decisions. In doing so, the framework removes the necessity to continually retrain RL agents in non-stationary and highly dynamic settings, while still maintaining adaptive and context-aware behavior. By providing empirical evidence that LLM-based controllers can match and in certain scenarios even exceed the performance of traditional MARL approaches in dynamic environments, this thesis opens up a new direction for adaptive control strategies, intelligent and flexible resource allocation, and scalable, distributed decision-making mechanisms in large, complex cyber-physical systems. Together, these contributions outline a cohesive and future-oriented framework for intelligent, energy-conscious, and explainable management of computational tasks in IoT networks. This framework not only optimizes how resources are allocated and used at the network edge, but also enhances transparency and adaptability, thereby establishing a solid basis for the next generation of flexible and autonomous edge intelligence systems.

1.3 Research Methodology

The proposed study employs a simulation-centered, system-level approach to design, build, and assess the XDT-FMARL framework. This approach integrates data-driven system modeling, the development of predictive DT, federated MARL, explainability and interpretability analysis, and LLM–supported decision processes into a unified end-to-end experimental workflow that ensures consistency from initial design through final evaluation.

The research begins with the preparation of both synthetic and real-world IoT datasets that capture temperature readings, CPU utilization, wireless latency, and battery discharge patterns, sourced from open repositories as well as from carefully controlled simulation runs. These datasets serve as the foundation for building a discrete-event IoT simulation testbed using the SimPy framework, which supports detailed and realistic emulation of device operations, task arrival processes, wireless communication dynamics, and system-level resource limitations. Within this simulated environment, a diverse set of device profiles is specified to represent heterogeneous hardware capabilities, distinct energy constraints, and a broad spectrum of network conditions, thereby mirroring the variability present in actual IoT deployments.

Subsequently, a dedicated DT layer is implemented to generate near-term forecasts of key system variables. For this purpose, lightweight prediction methods most notably linear regression models and moving-average–based estimators, were used to estimate and project, for each individual device, the battery charge level, processor utilization, and delay in wireless communication. Together, these DT outputs provide a contextual, forward-looking snapshot of the anticipated operating conditions of the system, shifted in time relative to the present. They are continuously updated as fresh sensor data and performance indicators arrive, ensuring that projections remain current. The revised predictions are then passed as input features to the learning components and decision-making units of the framework, where they inform and steer processes such as adaptive behavior, performance optimization, and closed-loop control actions.

Each IoT device is represented as an independent RL agent embedded in a larger multi-agent system. These agents are trained with a Dueling Double Deep Q-Network (DQN) that incorporates Prioritized Experience Replay (PER), which supports more robust and stable convergence even when the environment is non-stationary and only partially observable. The agent's action space includes both decisions to process tasks locally and decisions to offload computation to an edge server, enabling the learned policies to jointly optimize energy usage, end-to-end delay, and processing load distribution. To synchronize learning across the network

of devices while maintaining a decentralized architecture, a FL scheme is adopted, in which local policy parameters are periodically uploaded and combined via federated averaging to produce a shared global model that is then redistributed to the agents.

To improve both transparency and interpretability, the framework incorporates a dedicated explainability layer. Within this layer, gradient-based saliency techniques and feature importance analyzes are employed to systematically measure how strongly critical system variables, such as battery charge level, processor load, and communication latency affect offloading choices. The resulting explanations clarify the behavior of the learned policies, revealing which factors drive particular decisions and helping to verify that the decision-making process remains reasonable and consistent across diverse operating conditions.

In addition to the RL component, the framework integrates an LLM-powered decision intelligence module. A lightweight LLM receives structured prompts that encode DT forecasts, up-to-date device resource states, and metrics reflecting the convergence status of FL. Leveraging this extensive contextual information, along with few-shot examples distilled from the observed behavior of the MARL agents, the LLM produces real-time task offloading recommendations. These proposed actions are then rigorously evaluated against the decisions generated by the RL agents, enabling an examination of their mutual alignment, the LLM's responsiveness to dynamic environmental changes, and the overall effectiveness, reliability, and robustness of the final offloading policies.

Finally, the proposed system is evaluated through extensive, simulation-driven benchmarking, where it is compared against centralized single-agent RL, and non-federated multi-agent reference methods. System performance is evaluated through a wide range of indicators, such as energy preservation, task completion delay, task offloading rate, reward convergence patterns, explainability consistency, and the degree to which LLM-generated decisions match ground-truth actions. By applying this comprehensive evaluation methodology under both stable and fluctuating network conditions, the study delivers a thorough and rigorous examination of the proposed framework. This, in turn, confirms its appropriateness and effectiveness for enabling energy-efficient, transparent, and explainable task offloading in IoT

settings, even in the presence of dynamic and potentially unreliable connectivity.

1.4 Thesis Overview

This thesis consists of eight chapters. The organization of the thesis is designed to provide a systematic progression from the background concepts and related work to the proposed framework, experimental evaluation, and final conclusions.

Chapter 1 introduces the research context, motivation, objectives, hypotheses, and general structure of the thesis.

Chapter 2 reviews the relevant literature on task offloading, FL, DTs, RL, XAI, and LLM-based decision-making approaches in IoT environments, and identifies the key research gaps addressed in this work.

Chapter 3 presents the research domain and system model, including the DT-enabled IoT architecture, data sources, simulation environment, and the modeling assumptions underlying the proposed framework.

Chapter 4 introduces the theoretical foundations of the methods employed in this thesis, including FL, MARL, DT–based predictive models and explainability techniques.

Chapter 5 describes the proposed XDT-FMARL framework in detail, including its architecture, learning mechanisms, DT integration, explainability layer, and LLM-assisted decision intelligence.

Chapter 6 presents the simulation setup and evaluation methodology, including simulation design, performance metrics, baseline methods, and benchmarking procedures.

Chapter 7 reports and discusses the simulation results, analyzes the performance of the proposed framework under different network and workload conditions, and compares it with the baseline approaches.

Chapter 8 concludes the thesis by summarizing the main contributions and outlining the directions for future research.

Chapter 2

Literature Overview

2.1 Task Offloading and Resource Management in IoT and Mobile Edge Computing

As IoT environments grow larger and more intricate, traditional centralized computing models increasingly fail to meet strict requirements on latency, energy usage, and bandwidth, underscoring the importance of intelligent task offloading and coordinated resource management. Computation offloading denotes the delegation of computational workloads from devices with limited processing, energy, or connectivity resources to more capable platforms, such as edge servers or remote cloud infrastructures. Mach and Běcvář [29] present a detailed architectural and conceptual framework for computation offloading in Mobile Edge Computing (MEC), in which they organize offloading decisions along three core questions: *what* portions of an application should be offloaded, *where* these computations should be carried out within the network hierarchy, and *when* the offloading operation should be triggered. In their survey, they systematically examine the inherent trade-offs among execution delay, energy demands, and communication overhead, and, in doing so, propose a classification scheme that has become a reference point and is still extensively used in later work on IoT and MEC systems.

Extending this architectural viewpoint, Mao et al. [30] provide a foundational survey that jointly considers communication, computation, and caching within MEC environments. They argue that computation offloading should not be addressed as

a standalone issue, but as one component of a tightly integrated system where radio resource allocation, task scheduling, and edge server capacity must be co-optimized. This survey is widely referenced in the IoT community for clearly articulating the strong interdependence between networking and computation, and for revealing system-wide bottlenecks that emerge when offloading decisions are made without accounting for underlying resource management constraints.

More recent surveys broaden this perspective by delivering detailed examinations of computation offloading strategies in large-scale MEC environments. Feng et al. [31] provide a thorough summary of both optimization-based and learning-driven offloading techniques, emphasizing goals such as latency reduction, energy savings, and quality-of-service assurance. Their work underscores the shortcomings of static and heuristic policies in highly dynamic IoT settings and shows that adaptive offloading schemes can markedly enhance long-term system performance. Along the same lines, Dong et al. [32] introduce a refined taxonomy of task offloading strategies, classifying current approaches into reactive, predictive, and intelligent categories. Their study further identifies open research problems concerning scalability, heterogeneity, and real-time adaptability in emerging IoT infrastructures.

Resource management has become a key pillar of robust offloading frameworks, since contention for scarce edge resources directly affects both the viability of offloading and the overall stability of the system. Zhang et al. [33] offer an extensive survey of MEC resource management strategies, including computation task scheduling, bandwidth distribution mechanisms, and integrated joint optimization approaches that coordinate multiple resources simultaneously. Their findings highlight that offloading decisions cannot be made in isolation: they must be tightly coupled with the underlying resource allocation and provisioning policies to prevent bottlenecks, congestion, and subsequent performance deterioration. This coordination is especially critical in large-scale, dense IoT environments characterized by heterogeneous, time-varying workloads and stringent quality-of-service requirements.

In parallel, RL has attracted growing interest as a powerful approach for handling the complexity and uncertainty that characterize IoT resource management. Luo et al. [34] provide a survey of RL-based computation offloading strategies and

highlight their capability to autonomously learn adaptive policies without relying on explicitly defined system models. Although RL-driven methods deliver notable advantages in terms of flexibility, scalability, and the ability to cope with dynamic environments, the survey also points out several important drawbacks, such as substantial training overhead, slow convergence to effective policies, and challenges in interpretability and transparency of the learned behavior. These shortcomings, in turn, are encouraging active research on hybrid architectures that integrate learning-based decision mechanisms with predictive analytics and model-driven components, aiming to balance performance gains with practicality, robustness, and explainability in real-world IoT deployments.

Overall, prior research provides a solid basis for computation offloading and resource management in IoT and MEC environments. At the same time, it highlights ongoing issues with adapting to highly dynamic conditions, coordinating mechanisms across multiple architectural layers, and embedding predictive, intelligence-driven capabilities into the system. These gaps motivate the development of more advanced, integrated frameworks that can simultaneously handle offloading decisions, allocate and orchestrate resources, and maintain holistic system-level awareness within a single, unified approach.

Local Versus Edge and Cloud Execution

Early work on computation offloading mainly examined the balance between local execution and remote processing. Mao et al. [35] proposed a queueing-theoretic and Lyapunov-based framework for dynamic offloading that shows how to trade off energy and delay under stochastic workloads. Although it provides theoretical guarantees, it is reactive and relies on simplified system models.

Recent work extends this binary view into an edge–cloud continuum. Pournazari et al. [36] show that modern IoT systems span multiple tiers—devices, edge servers, and cloud platforms and that execution placement should consider not only computational power but also network conditions, service needs, and scalability, motivating more flexible offloading than simple local-versus-cloud choices.

Reactive Versus Proactive Offloading Strategies

Reactive offloading schemes base their decisions only on the instantaneous system state, which may result in suboptimal performance in rapidly changing environments. To overcome this drawback, Zhang et al. [37] introduced proactive association and offloading approaches grounded in Markov decision processes and Lyapunov-based control, allowing the system to anticipate future states. Their findings show that such anticipatory decision-making can markedly enhance long-term performance relative to purely reactive strategies.

Digital Twin–Assisted Proactive Offloading

DT technologies have recently become a key enabler of proactive task offloading by offering predictive insight into system behavior. Prior work shows that DT-assisted offloading enhances decision accuracy by predicting resource availability, network latency, and workload levels. DT-based offloading has been effectively used in vehicular edge computing and multi-service IoT, where virtual counterparts of physical systems enable anticipatory task assignment [38, 39]. These studies directly compare reactive schemes with DT-supported proactive methods, emphasizing the advantages of prediction-driven decisions.

Joint Offloading and Resource Management

Effective offloading is inseparable from resource management, especially in mission-critical IoT systems. Rasouli et al. [40] show that latency and reliability requirements demand joint optimization of computation placement and resource allocation. Deng et al. [41] further demonstrate that intermittent connectivity and network uncertainty can significantly harm purely reactive offloading policies, highlighting the need for adaptive, predictive frameworks that coordinate offloading and resource management under realistic conditions.

2.2 Federated Learning for Distributed IoT Intelligence

The growing scale, diversity, and data volume of IoT systems have made conventional centralized machine learning (ML) increasingly impractical, mainly because of high communication costs, privacy risks, and poor scalability. FL has arisen as a decentralized approach that enables multiple distributed devices to jointly train models while keeping their raw data on-device. This characteristic makes FL especially well-suited to large IoT environments, where devices have limited resources, data are non-identically distributed, and network conditions change frequently.

2.2.1 Fundamentals of Federated Learning in IoT

FL was first proposed as a distributed optimization paradigm for edge devices, in which a global model is iteratively updated by aggregating locally computed model updates instead of collecting raw data centrally. Li et al. [42] offer a basic introduction to FL, highlighting key challenges that include statistical heterogeneity, system heterogeneity, and communication efficiency. Within IoT environments, these issues become more pronounced due to sporadic connectivity, limited battery resources, and heterogeneous sensing modalities.

Nguyen et al. [43] provide one of the most comprehensive overviews of FL in the context of IoT systems, emphasizing its ability to support distributed intelligence while meeting privacy and scalability constraints. In their survey, they classify FL-based IoT applications in several key domains, including smart cities, industrial automation, healthcare, and vehicular networks. Through this systematic categorization and analysis, they underscore the role of FL as a fundamental technology for achieving decentralized, privacy-aware intelligence at the network edge.

2.2.2 Cross-Device Federated Learning and System Heterogeneity

In most IoT scenarios, FL is deployed in a *cross-device* setting, where training involves a large population of devices, each holding only a small, uneven share of

the data. This configuration naturally results in pronounced system heterogeneity, since participating devices vary in processing power, battery and energy constraints, as well as the stability and bandwidth of their communication links.

Recent studies increasingly underscore that traditional aggregation strategies, notably FedAvg, can experience slow convergence and reduced accuracy when faced with highly non-IID data distributions that are typical in IoT environments. In particular, Aggarwal et al. [44] and Alsharif et al. [45] provide systematic examinations of these shortcomings and outline a range of enhancements and design directions intended to strengthen robustness, scalability, and adaptability in heterogeneous IoT deployments.

2.2.3 Challenges in Dynamic IoT Environments

Dynamic IoT settings introduce further difficulties for FL-driven distributed intelligence. Variations in network quality over time, changing computational demands, and strict energy budgets can significantly degrade both training performance and long-term system viability. Liu et al. [46] emphasize that continual shifts in device participation and underlying data distributions may cause model drift and hinder stable convergence, especially when FL is applied in deployments that run over extended durations.

Moreover, recent literature highlights that FL systems need to move past fixed, one-size-fits-all training regimes and instead adopt adaptive client participation and context-sensitive coordination mechanisms. Ayeelyan et al. [47] contend that next-generation FL frameworks should embed both predictive capabilities and system-level intelligence to better cope with environmental variability while simultaneously minimizing training costs and communication overhead. At the same time, new research in industrial IoT underlines the necessity of FL approaches that are not only robust and reliable but also transparent and interpretable, in order to enable their safe adoption in mission and safety-critical application domains [48].

2.3 Digital Twins for Predictive IoT Systems

DTs are virtual replicas of physical entities that continuously track and reflect the condition and behavior of their real-world originals through ongoing, synchronized data exchange. Within IoT ecosystems, DTs are typically used to model devices, sensors, communication elements, and even complete infrastructures, thus sustaining a persistent link between physical operations and their digital models. Mihai et al. [49] present an extensive overview of the technologies that underpin DT and stress that uninterrupted synchronization and two-way data exchange are key features that separate DTs from traditional simulation approaches. Their survey further underscores that real-time data acquisition and high-fidelity modeling are crucial to preserving accurate and up-to-date virtual representations, especially in highly dynamic and rapidly changing environments.

Focusing on the Industrial IoT domain, Xu et al. [50] examine DT architectures and associated tool chains, highlighting multi-layer DT structures that span data acquisition, modeling, analytics, and control functionalities. They contend that DTs markedly improve system observability by aggregating diverse IoT data streams into a coherent virtual counterpart of the physical system. Building on this view, Hakiri et al. [51] review DT deployment in next-generation networks and emerging IoT sectors, emphasizing scalability constraints, synchronization delays, and interoperability issues as key architectural hurdles. Taken together, these works position DTs as a crucial system-awareness substrate that underpins intelligent, adaptive, and predictive IoT solutions.

Resource and Performance Forecasting Using Digital Twins

A key function of DT-enabled IoT systems is their capacity to anticipate resource usage and performance indicators over short- to medium-term time frames. Mihai et al. [49] highlight that DTs can combine historical data with continuous sensor streams to estimate factors such as battery lifetime, processing demand, and network quality. These predictive capabilities enable systems to adjust configurations in advance, which is especially important for managing limited resources and maintaining reliable operation in constrained IoT environments.

Xu et al. [50] further note that lightweight prediction approaches such as regression models and basic statistical trend analyses are well suited to edge-focused DT deployments because they incur minimal computational cost and can be executed on constrained devices. They caution, however, that the accuracy of such predictive models tends to deteriorate over time as workload characteristics and environmental factors shift, leading to model obsolescence if no countermeasures are taken. To mitigate this limitation, Li et al. [52] introduce a DT-based service provisioning framework that leverages continual learning, enabling DT models to be updated incrementally so that they track and adapt to ongoing changes in system behavior and operating conditions. Their empirical findings indicate that maintaining reliable forecasting capabilities in non-stationary IoT settings requires not only frequent synchronization between physical assets and their digital counterparts, but also ongoing, adaptive refinement of the underlying predictive models.

Hakiri et al. [51] conclude that when DT-based forecasting is sufficiently accurate, it supports proactive resource allocation and control, which in turn mitigates system instability and enhances overall energy efficiency, even when network conditions and workloads vary dynamically and unpredictably.

DT-Assisted Decision Making in IoT Systems

Beyond their use for monitoring and prediction, DTs are increasingly employed as decision-support tools for implementing intelligent control in IoT environments. Mihai et al. [49] emphasize that DTs facilitate what-if analysis by enabling prospective control strategies to be tested and assessed in the virtual replica before they are applied to the physical system. By experimenting with different scenarios, configurations, and parameter settings in this simulated space, operators can identify potential issues, refine control policies, and compare alternative options. As a result, this approach not only mitigates operational risk but also improves the robustness and reliability of system behavior in complex, dynamic, and safety-critical environments.

Within the domain of task offloading and resource orchestration, Zhang et al. [53] propose a DT-based intelligent task offloading framework tailored for collaborative

mobile edge computing. In their design, anticipated system conditions captured in the DT are exploited to steer offloading strategies, leading to notable reductions in latency and enhanced system robustness relative to conventional reactive schemes. Along the same lines, Tran-Dang and Kim [54] present a comprehensive, state-of-the-art survey on DT-enabled computation offloading in edge environments. They conclude that DT-supported decision processes, by embedding predictive contextual information into the offloading logic, reliably surpass purely reactive approaches in both performance and adaptability.

Li et al. [52] additionally demonstrate that coupling DTs with adaptive learning strategies can improve the quality of decisions over extended periods, as these mechanisms help preserve consistency between digital replicas and their corresponding physical assets. Nevertheless, Hakiri et al. [51] note that existing DT-based decision-making frameworks largely concentrate on optimizing performance metrics, while paying comparatively little attention to making decisions transparent and interpretable. This gap underscores a pressing need for DT-enabled control approaches that not only boost operational efficiency but also facilitate reliable, explainable, and thus more trustworthy decision processes in autonomous IoT environments.

2.4 Reinforcement Learning and Multi-Agent RL for Offloading

RL has emerged as a widely used and promising strategy for managing the complexity and uncertainty inherent in computation offloading decisions within IoT and MEC environments. In contrast to traditional optimization techniques that depend on accurate and often hard-to-obtain system models, RL allows agents to gradually derive effective offloading policies by continuously interacting with and observing the environment. This data-driven learning process makes RL particularly well-suited to highly dynamic and heterogeneous IoT scenarios, where devices, connectivity, and computing resources can change rapidly. Recent survey studies further indicate that RL-based offloading solutions show notable advantages in settings characterized by time-varying network conditions, fluctuating workloads, and changing resource

availability, consistently achieving robust and adaptive performance across diverse operational contexts.[55].

Single-Agent Reinforcement Learning for Offloading

Early RL-based offloading methods mainly adopt a single-agent framework, in which either a centralized controller or an individual agent on each device monitors the current system conditions and then selects whether tasks should be processed locally or transferred to edge or cloud servers for execution. In their survey, Luo and Dai [34] present an in-depth review of these methods, highlighting that both traditional Q-learning and more advanced deep reinforcement learning (DRL) algorithms are capable of achieving a favorable trade-off between latency and energy consumption, while avoiding the need for precise analytical system models. They further emphasize that such data-driven approaches can adapt to dynamic environments and heterogeneous devices, making them particularly suitable for practical mobile and edge computing scenarios.

Method-level research in recent years has further confirmed that single-agent DRL can be effectively applied in dynamic MEC scenarios. Xie and Cui [58] design a DRL-based dynamic offloading policy that continuously adjusts to time-varying task arrivals and wireless channel conditions, and they show that this adaptive approach achieves a superior delay–energy balance relative to conventional static schemes. In a related line of work, Fan et al. [59] cast the joint offloading and server selection problem as a Markov decision process and utilize DRL to handle rapidly changing network states, demonstrating significant improvements in execution delay when compared with baseline methods.

Despite their advantages, single-agent RL methods come with fundamental drawbacks. As emphasized by Hortelano et al. [55], a centralized decision-making paradigm scales poorly as the number of IoT devices grows and often overlooks the competitive interactions among devices contending for the same edge resources. In addition, single-agent formulations generally rely on globally observable environments or demand rich, detailed state information. Such requirements are difficult to satisfy in large, heterogeneous, and privacy-sensitive IoT deployments, where full system vis-

ibility is limited and sharing fine-grained data may not be feasible or even allowed by regulatory constraints.

Multi-Agent Reinforcement Learning for Cooperative Offloading

To overcome limitations in scalability and coordination, recent work is increasingly focusing on MARL. In this framework, multiple agents simultaneously learn distributed policies while interacting in a common environment. When applied to offloading, each IoT device or edge node is modeled as an independent agent that makes and refines its own local decisions while still coordinating either implicitly through shared dynamics or explicitly through communication with other agents in the system.

Yao et al. [60] study the use of MARL for task offloading in crowd-edge computing environments and show that, even under partial observability, decentralized agents can jointly attain system performance that is close to the global optimum. Their findings further indicate that MARL offers markedly better scalability than centralized RL approaches, especially in large-scale, dense, and highly heterogeneous network settings. Along the same lines, Xiong et al. [61] develop a multi-agent DRL framework equipped with explicit coordination mechanisms for distributed task offloading, underscoring that effective cooperation among agents is crucial for lowering execution latency, alleviating resource contention, and improving overall system efficiency.

Systematic reviews further corroborate that MARL is particularly appropriate for large-scale IoT scenarios, as its decentralized structure and intrinsic parallelism align well with such distributed environments [57]. At the same time, this work highlights several persistent challenges, most notably the non-stationarity arising from multiple agents learning concurrently, the substantial computational and algorithmic complexity of training, and the tendency toward slow convergence when network conditions fluctuate rapidly or unpredictably.

2.5 Explainable Artificial Intelligence in Distributed Systems

2.5.1 Explainability Concepts in Distributed and Autonomous Systems

XAI has become a key area of research aimed at improving the transparency, accountability, and trustworthiness of ML models, with particular importance in distributed and autonomous systems. Barredo Arrieta et al. [62] present an extensive classification of XAI approaches, differentiating between models that are inherently interpretable and post-hoc explanation methods that are applied to opaque black-box learners. Their review underlines that explainability is strongly dependent on the specific context of use and should be adapted to the underlying system architecture, the nature of the decision-making workflow, and the distinct needs and expectations of the various stakeholders involved.

In distributed systems, explainability extends beyond the interpretation of single models and requires reasoning at the level of the entire system, where overall outcomes arise from the interactions among many agents, devices, and services. Guidotti et al. [63] argue that explanations for black-box models should illuminate not only the final predictions, but also the internal decision logic that produces them, which is especially important in scenarios where these decisions affect how resources are distributed or how system robustness and stability are preserved. From a practical and operational perspective, Gunning and Aha [64] further argue that explainability is a key prerequisite for effective human–AI teaming, particularly in autonomous or semi-autonomous systems in which human operators must be able to inspect, verify, and audit machine outputs, as well as develop appropriate trust and confidence in AI-driven decisions over time.

These conceptual foundations are especially important in distributed IoT and edge computing environments, where control and decision processes are decentralized and must adapt to heterogeneous, dynamically changing system conditions. In these settings, explainability functions as a key instrument for connecting autonomous optimization procedures with human understanding and governance, en-

suring that system behavior remains transparent, interpretable, and open to meaningful oversight and intervention.

2.5.2 Gradient-Based Explanation Methods for Distributed Learning Models

Gradient-based explanation methods are commonly used to interpret deep learning models because they are computationally efficient and can be easily integrated into large-scale production pipelines. Sundararajan et al. [65] propose integrated gradients, a principled attribution technique that quantifies feature importance by integrating the gradients along a continuous path from a chosen baseline input to the actual input instance. This method is designed to satisfy important axiomatic properties, including sensitivity and implementation invariance, which provide theoretical guarantees about the reliability of the resulting attributions. Consequently, integrated gradients is particularly well-suited for analyzing and explaining the decision boundaries of deep neural networks, including those deployed in resource-constrained edge environments, where transparent and trustworthy model behavior is essential.

Shrikumar et al. [66] introduce Deep Learning Important Features (DeepLIFT), an attribution technique that explains model predictions by contrasting neuron activations with those obtained from a chosen reference input, instead of depending exclusively on local gradient information. By doing so, DeepLIFT mitigates the problem of gradient saturation and yields explanations that are typically more robust and reliable, a property that is particularly valuable for deep architectures frequently deployed in distributed intelligence pipelines and multi-stage inference systems.

For convolutional or more generally spatially structured models, Selvaraju et al. [67] propose Gradient-weighted Class Activation Mapping (Grad-CAM), which utilizes the gradients flowing back to convolutional feature maps to construct class-specific, discriminative localization heatmaps. These visualizations highlight the regions of the input that most strongly influence a particular prediction. Grad-CAM has been widely adopted to interpret and visualize decision-making behavior

in edge-based inference, perception modules in embedded systems, and a range of real-time computer vision tasks.

The reliability of gradient-based explanations has also been critically examined. Adebayo et al. [68] conduct extensive sanity checks on saliency methods and demonstrate that some explanation techniques may produce misleading attributions if not properly validated. Their findings highlight the need for robust explanation mechanisms, particularly in distributed and safety critical systems where erroneous explanations can undermine trust.

2.5.3 Transparency and Accountability in Distributed Decision Making

Trust and transparency are essential prerequisites for the deployment of intelligent decision-making systems in distributed settings, including IoT infrastructures, FL frameworks, and multi-agent control systems. To support these requirements, model-agnostic explanation methods such as SHapley Additive exPlanations (SHAP), proposed by Lundberg and Lee [69], and Local Interpretable Model-agnostic Explanations (LIME), introduced by Ribeiro et al. [70] provide local interpretability by either assigning contributions of individual input features to model predictions or by approximating complex models with simpler, human-interpretable surrogates. In practical applications, these techniques are widely adopted to justify and analyze decisions that depend on operational metrics like battery charge status, processor utilization, queue occupancy, and communication or network latency, thereby making the behavior of distributed intelligent systems more understandable and auditable.

In distributed learning settings, however, explainability poses further complications. Nguyen et al. [71] demonstrate that explanation methods can unintentionally reveal sensitive information, underscoring the need for privacy-preserving and security-aware explainability techniques in decentralized environments. In addition, Dubey et al. [72] point out that combining XAI with FL and distributed optimization demands careful orchestration of system components, since explanations must be produced without relying on centralized access to raw data, full model

parameters, or complete global system states, while still ensuring that the resulting explanations remain reliable and informative.

2.6 Comparative Analysis of Existing Approaches

2.6.1 Identified Research Gaps

The reviewed literature establishes FL as a strong foundation for distributed intelligence in IoT systems, primarily due to its scalability and its ability to preserve data locality, thereby avoiding many limitations of centralized learning architectures. Nevertheless, several key issues remain open. In particular, device and network heterogeneity, time-varying execution conditions, and the coordination overhead of large-scale federated systems still constrain the robustness and efficiency of FL-based IoT deployments. These limitations suggest that FL by itself cannot ensure sustained, adaptive intelligence in dynamic, resource-limited environments.

RL and MARL have been widely explored as solutions for computation offloading and resource management in IoT and edge environments. Although these methods show considerable promise, most RL-based offloading schemes depend on reactive decisions based solely on current observations. Consequently, they are unable to proactively anticipate future changes in resource availability, workload fluctuations, and network dynamics, which are typical in real-world IoT settings. This limitation can lead to performance deterioration when operating conditions change rapidly. While MARL enhances scalability by distributing control among multiple agents, it also brings significant challenges in terms of training complexity, convergence reliability, and communication overhead, thereby hindering practical deployment in large-scale, energy-constrained networks.

Another limitation frequently noted in the literature is the weak incorporation of predictive system awareness into learning-based offloading schemes. As recent surveys indicate [34, 55], RL and MARL policies are rarely combined with predictive models that estimate short-term system evolution. Although DTs have proven effective in forecasting resource conditions such as battery status, computational load, and network latency, they are predominantly used for monitoring or offline simu-

lation. Their systematic use in real-time offloading decision-making is still scarce, leading to a shortage of DT-aware, forward-looking offloading strategies.

Furthermore, the interpretability and transparency of autonomous offloading strategies remain insufficiently studied. Most RL- and MARL-based methods focus on optimization goals such as minimizing latency or improving energy efficiency, yet offer limited visibility into the reasoning behind specific decisions. This lack of explainability undermines trust, verification, and accountability, especially in mission-critical IoT scenarios, where it is crucial to understand the behavior of the system. Although XAI methods have advanced considerably, their integration into distributed offloading and FL architectures is still sparse and frequently considered an add-on rather than a core design element.

Finally, the existing literature shows a lack of advanced contextual reasoning mechanisms that can adapt decision-making strategies without requiring substantial retraining. LLMs, despite their strong in-context learning and reasoning abilities, remain largely unexamined as decision-support modules for IoT offloading and resource management. Current solutions usually rely on frequent RL retraining or federated model updates to respond to environmental changes, which introduces considerable overhead and slows down adaptation. The absence of LLM-driven reasoning that leverages DT forecasts, historical decision trajectories, and a broader system context constitutes a major open research challenge.

Taken together, these limitations emphasize the need for unified architectures that couple FL with predictive DTs, strengthen RL and MARL through proactive system awareness, incorporate explainability for transparent decision-making, and exploit LLM-based contextual reasoning to boost adaptability while lowering training complexity. Overcoming these challenges is crucial for the realization of robust, scalable, and reliable distributed intelligence in next-generation IoT systems.

2.6.2 Strengths and Weaknesses of Existing Approaches

Table 2.1 shows that current methods tackle individual facets of distributed IoT intelligence but do not deliver a comprehensive solution. Centralized and heuristic approaches face scalability and adaptability constraints, and although FL enhances data decentralization, it does not inherently support real-time decision making. RL-based offloading adds adaptivity but remains mostly reactive and computationally intensive, especially in multi-agent scenarios. DTs enable predictive analysis yet are seldom embedded into operational control loops, while explainability is frequently incorporated only as a post-hoc layer rather than a fundamental design element. Recent LLM-based techniques exhibit strong contextual reasoning but are still largely unexplored in resource-limited IoT settings. Together, these insights underscore the need for an integrated framework that unifies predictive DTs, FL, adaptive multi-agent decision making, and explainability within a single cohesive architecture.

Table 2.1: Strengths and Weaknesses of Existing Approaches for Distributed IoT Intelligence

Approach	**Strengths**	**Weaknesses**
Centralized Processing	Global system visibility; simple coordination and control; mature optimization techniques	High latency; network congestion; single point of failure; privacy concerns; poor scalability for large IoT deployments
Edge-Based Offloading (Static / Heuristic)	Reduced latency compared to cloud; simple implementation; low computational overhead	Inflexible to dynamic conditions; suboptimal resource utilization; lacks learning and adaptation
Federated Learning	Preserves data locality; scalable training across heterogeneous devices; reduced communication of raw data	Sensitive to device heterogeneity; communication overhead; lacks runtime adaptivity and decision intelligence
Reactive RL-Based Offloading	Learns adaptive policies from interaction; improves energy and latency trade-offs over heuristics	Purely reactive; slow convergence; unstable under non-stationary conditions; high training cost
Multi-Agent RL	Distributed decision making; improved scalability; localized control	Complex coordination; convergence instability; communication overhead; limited practical deployability
DT–Based Monitoring	Accurate system state representation; predictive capability for resources and performance	Often used offline; weak coupling with real-time decision making; limited integration with learning policies
XAI for Offloading	Improves transparency and trust; supports debugging and validation	Mostly post-hoc; rarely integrated into control loops; explanation cost not considered
LLM-Based Reasoning	Strong contextual reasoning; rapid adaptation via in-context learning; minimal retraining	Limited exploration in IoT; integration challenges; explainability and control guarantees still open

2.6.3 Justification of Research Direction

The comparative analysis in the previous sections, together with the comparative overview in Table 2.1, shows that current methods for distributed IoT intelligence each tackle different, largely disconnected facets of the overall challenge. Centralized and heuristic-based strategies face limitations in scalability and adaptability, whereas FL decentralizes training but offers no native support for real-time decision making or runtime awareness. RL and MARL add adaptivity for computation offloading, yet they are mainly reactive, costly to train, and vulnerable to non-stationary conditions. In addition, although DT techniques provide strong predictive capabilities, they are usually not embedded within operational control loops, and explainability is seldom treated as a core element of distributed decision-making systems.

These observations highlight the need for a unified research agenda that goes beyond incremental refinements of isolated techniques. Specifically, there is strong motivation to integrate FL with adaptive offloading strategies that are proactive and system-aware, allowing IoT devices to anticipate future resource conditions instead of responding only to instantaneous states. Incorporating DTs offers a systematic way to embed predictive system awareness by forecasting key operational metrics such as battery dynamics, computational demand, and network latency. When these forecasts are integrated directly into the decision-making pipeline, they can improve the robustness and efficiency of learning-based offloading policies in dynamic environments.

Furthermore, the complexity and convergence issues associated with MARL-based offloading motivate the investigation of complementary decision-support approaches that lower training costs while maintaining adaptability. In this regard, LLMs provide a promising framework for contextual reasoning, as they can exploit structured system summaries, historical trends, and predictive information to guide offloading decisions without the need for continuous retraining. At the same time, the limited interpretability of autonomous offloading policies calls for the incorporation of XAI methods to guarantee transparency, trust, and accountability particularly in safety-critical or resource-constrained IoT environments.

Accordingly, this thesis pursues a research direction focused on designing and validating an integrated framework that unifies FL, predictive DTs, adaptive MARL, and explainability mechanisms within a single coherent architecture. By jointly addressing predictive awareness, distributed learning, adaptive decision-making, and interpretability, the proposed approach seeks to overcome the limitations of existing solutions and advance robust, scalable, and reliable distributed intelligence for next-generation IoT systems. To examine and implement this methodology, the next chapter introduces the system modeling foundations of the proposed framework. It specifies the network architecture, device assumptions, state representations, communication model, and optimization objectives that support the integration of FL, DT, MARL and XAI and allow for an assessment of the offloading strategy.

Chapter 3

System Architecture of the Proposed Framework

This chapter presents the architectural foundations and underlying system conditions of the IoT environment examined in this thesis. It concentrates on the structural organization of devices, their resource limitations, data generation patterns, and communication properties, all of which jointly define the operational setting for distributed intelligence. The detailed mechanisms for intelligent decision-making and optimization are analyzed in Chapter 5.

3.1 IoT System Model

The IoT system is modeled as a distributed network of diverse devices with resource constraints that act autonomously and communicate over a shared wireless medium. Each IoT node is treated as an independent processing unit with sensing, computing, and communication functions. This abstraction captures the decentralized and asynchronous behavior of practical IoT deployments, where devices vary widely in hardware resources, energy budgets, and workload profiles.

Device heterogeneity is captured explicitly via variations in computing capacity, energy reserves, and communication conditions. The processing power of each device is represented by its available CPU resources, which determine both the execution latency and the energy consumed by locally executed tasks. IoT devices are

considered to operate under strict energy constraints as they typically depend on batteries or energy-harvesting mechanisms. Consequently, battery depletion emerges as a critical limiting factor that directly influences long-term system viability and decision-making. Energy is spent on both local computation and wireless transmission, and battery levels are continuously updated to reflect the accumulated impact of these operations over time.

The generation of tasks on each IoT device is represented as a stochastic process designed to reflect realistic sensing and application workloads. The produced tasks vary in input size, computational demand, and latency requirements, thus covering various application domains such as industrial monitoring, vehicular sensing, and healthcare data processing. Each task is associated with a deadline that imposes strict latency constraints, highlighting the need to carefully trade off local execution against offloading to remote resources. Any task that does not complete before its deadline is labeled unsuccessful, directly influencing the system's overall performance metrics.

Communication between IoT devices and edge infrastructure is represented as a time-varying wireless network with stochastic latency. Network delay is modeled as a dynamic parameter that changes with channel conditions, background traffic, and occasional congestion. This variability captures realistic IoT settings, where latency is neither fixed nor fully predictable. Consequently, communication delay becomes a key constraint that affects task completion time and the viability of task offloading.

Overall, the IoT system model highlights the interaction between diverse device capabilities, constrained energy budgets, computational limitations, and time-varying communication conditions. Together, these factors create a demanding operational setting in which intelligent task execution and resource management approaches must be assessed. By making these constraints explicit, the proposed system model offers a realistic and adaptable basis for examining adaptive offloading and distributed intelligence techniques in the following chapters.

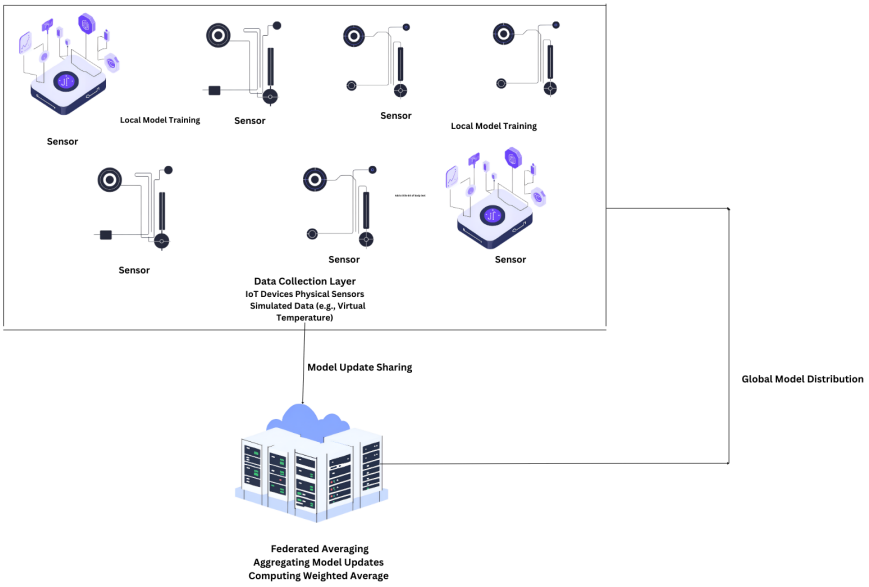


Figure 3.1: Federated learning–enabled IoT system architecture.

3.1.1 Federated Learning–Enabled IoT System Architecture

Beyond local sensing and computation, the IoT system model in this thesis is also built to enable distributed intelligence via FL. Each IoT device acts as an independent learning node, training models locally on data produced by its own sensors. This approach maintains data on the device and mirrors practical deployments where raw sensor data cannot be centrally aggregated because of privacy, bandwidth, or energy limitations.

As shown in Fig. 3.1, IoT devices periodically train local models on their own private data and send only the resulting model updates to an edge or cloud aggregation server. This server performs FedAvg via weighted averaging to build a global model that integrates knowledge from diverse devices. The updated global model is then sent back to all participating nodes, enabling joint learning without sharing raw data.

This FL-based architecture provides a scalable and privacy-conscious backbone for distributed intelligence in the IoT ecosystem. Clearly incorporates device heterogeneity, recognizing that nodes can vary in sensing types, processing power, energy reserves, and communication reliability. In addition, regular synchronization between local and global models creates system-level dynamics that interplay with resource limits such as battery usage, CPU load, and network delay.

Within the broader system model, FL serves as a core mechanism that supports higher-level decision-making. In the following sections, predictive DTs, adaptive offloading schemes, and intelligent control methods leverage this FL-based infrastructure to improve energy efficiency, reduce latency, and increase system robustness under dynamic operating conditions.

3.2 Adaptive Task Offloading in IoT Systems

This thesis examines a DT–enabled IoT architecture designed to facilitate predictive, resource-efficient, and learning-oriented operation under the typical constraints of large-scale IoT systems. The architecture follows a layered design in which physical sensing and edge computation are supported by continuously updated DT models, while task execution and learning processes are managed by an intelligent decision layer and a federated aggregation scheme. This layering offers modularity and allows for the systematic integration of heterogeneous devices, predictive models, adaptive control, and distributed learning.

3.2.1 Physical Sensing and Local Execution Layer

The physical sensing layer is composed of various IoT devices that integrate sensors with limited local computation and communication resources. Each device functions autonomously and is represented as an independent process, capturing the inherently asynchronous nature of real-world IoT systems. These devices generate computational tasks from both sensed data streams and higher-level applications, with workloads varying in input size, processing complexity, and latency requirements. In the simulation framework used in this thesis, the input sizes of the tasks range from 64 to 512 KB, the computational requirements span 0.5 to 3.0 GFLOPs per task, and the deadlines range from 150 to 400 ms [56] to approximate delayed services.

Resource limitations at this layer are explicitly represented. Each device is modeled with finite battery energy, constrained CPU capacity, and time-varying wireless connectivity. Battery exhaustion is treated as the main sustainability constraint,

with energy consumption arising from both computation (local task execution and learning) and communication (transmitting model updates and offloading-related control traffic). The CPU load dictates achievable task processing rates and directly influences energy usage. Wireless network latency is captured as a stochastic, time-varying process to reflect both typical conditions and sporadic congestion. Specifically, latency samples are drawn around an average of 30 ms, with low probability extra delays of 5–10 ms [56] added to emulate transient network disruptions. Collectively, these aspects define a demanding operating environment in which devices must continually trade off performance objectives against constrained resources.

3.2.2 Digital Twin Layer

In the proposed architecture, each physical IoT node is linked to a lightweight DT that preserves a continuously updated virtual model of the device and its operating environment. As shown in Fig. 3.2, the DT layer ingests real-time telemetry from sensor nodes, such as battery status, CPU load, and observed wireless latency. These raw inputs are initially handled by a data aggregation and normalization module, which enforces temporal consistency and converts heterogeneous measurements into structured state variables appropriate for subsequent analysis.

Based on aggregated data, the DT applies predictive analytics to estimate short-term trends in resource availability and system performance. Lightweight forecasting methods, including moving averaging and regression-based estimators trained on simulated traces, are used to predict near-future changes in battery usage, processing load, and communication latency. With these forecasts, the DT moves beyond passive monitoring and offers anticipatory insight into the short-term behavior of the system over the next decision interval.

The outputs of the DT predictive analytics module are exposed to the decision intelligence layer via structured queries, as illustrated in Fig. 3.2. Instead of depending only on instantaneous measurements, offloading decisions draw on both the device's current operating state and its predicted resource availability. This approach enables proactive task execution strategies that mitigate performance degradation caused by sudden workload variations, network congestion, or energy consumption.

Within the overall system architecture, the DT layer serves two mutually reinforcing functions. First, it improves situational awareness by integrating short-term prediction directly into the control loop, thereby increasing robustness in the presence of non-stationary workloads and fluctuating network conditions. Second, it provides an abstraction boundary between low-level sensing and high-level decision making by compressing raw telemetry into a concise and interpretable set of state variables and forecasts. This abstraction enables learning-based controllers to remain lightweight and scalable while still exploiting predictive context to enable intelligent and energy-efficient decisions.

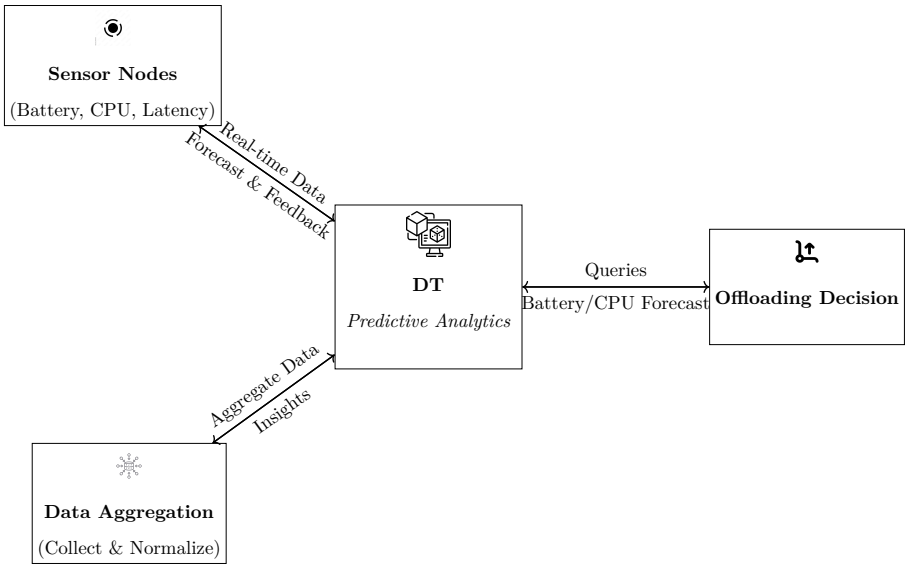


Figure 3.2: DT Architecture

3.2.3 Adaptive Offloading Interface

A key feature of the proposed IoT architecture is adaptive computation offloading, allowing devices to dynamically decide whether to process tasks locally or delegate them to edge resources. This feature is implemented as an architectural interface within the decision intelligence layer and is explicitly tailored to use predictive insights from the DT layer.

At each decision point, an IoT device must choose between two execution options: handling incoming tasks locally with its own computing resources or offloading them to an edge server over the wireless network. This decision is adaptive rather than fixed, mirroring the highly dynamic nature of IoT settings where resource

availability, workload levels, and network conditions change over time. The offloading interface thus serves as a bridge between the physical execution layer and the higher-level decision-making logic.

Unlike purely reactive offloading approaches that depend only on instantaneous measurements, the adaptive offloading interface takes advantage of short-term predictions produced by the associated DT. These predictions offer early warnings of battery drain, CPU overload, and communication latency. Using expected rather than solely current system conditions, the architecture supports proactive offloading decisions that alleviate emerging resource constraints before they fully develop.

From an architectural standpoint, the adaptive offloading interface does not dictate how decisions are derived; instead, it specifies what information is available and where decisions are enacted. It receives structured state variables and forecasts from the DT layer, together with coordination signals from the federated aggregation layer, and produces a standardized decision output that determines where tasks will be carried out. This separation allows a wide range of decision-making methods, including rule-based policies, learning-driven controllers, and reasoning-focused mechanisms, to be integrated without modifying the core system model.

By treating adaptive offloading as a core architectural element, the proposed system enables a shift from purely reactive execution toward predictive, context-aware task handling. This approach guarantees that offloading choices are systematically based on both the current limitations of the local device and the expected behavior of the broader system, while preserving architectural flexibility to incorporate the advanced decision-making mechanisms developed in subsequent chapters.

3.2.4 Decision Intelligence Layer

The decision intelligence layer specifies the architectural interface used to determine where tasks are executed within the IoT system. Its main function is to offer a structured way to choose between running tasks locally or offloading them to edge resources, according to each device's operating context. Rather than enforcing a particular decision-making method, this layer defines the location and manner in which execution decisions are integrated into the system.

In the modeled system, execution decisions are made in discrete time steps. Before each decision point, an IoT device refreshes its local state description, which includes metrics such as remaining battery charge, current CPU load, measured wireless latency, and the fill level of its local task queue. These state variables are provided by the physical sensing layer and augmented with predictive data from the system awareness and prediction layer.

A key characteristic of this layer is that decisions are based on contextual system information rather than relying solely on instantaneous measurements. Short-term forecasts from the prediction layer provide advance insight into imminent resource conditions, and coordination signals from the FL process supply a global view of the system. By revealing both current and expected system states, the decision intelligence layer supports execution placement over a wider time horizon, without imposing any assumptions about how this information is interpreted.

From an architectural perspective, decision making is distributed among IoT nodes, aligning with the inherently decentralized nature of large-scale IoT systems. Each node enforces execution decisions locally, yet supports periodic system-wide coordination via information exchange and aggregation. The layer is deliberately kept independent of any specific decision method, enabling diverse approaches, such as learning-based controllers or reasoning-driven decision support, to be incorporated and assessed within a unified system model.

By clearly separating decision interfaces from decision logic, this layer establishes a modular control point that captures the interaction between computation, communication, and energy constraints. This abstraction offers a realistic and flexible basis for the adaptive offloading mechanisms developed and examined in later chapters, while avoiding optimization-specific assumptions in the architecture itself.

3.2.5 Federated Aggregation Layer

The federated aggregation layer specifies the system-level process that coordinates distributed learning across the IoT network while keeping data on the originating devices. In this setup, each IoT node acts as an independent learner, training models on locally collected sensor data. Raw data never leave the device, only

model parameters or updates are transmitted to the aggregation infrastructure.

At predefined synchronization points, devices send their locally computed model updates to an aggregation server that operates at the edge or in the cloud. This server merges updates using a federated aggregation method, such as FedAvg, to form a global model that captures information from the entire distributed network. The updated global model is then sent back to the participating devices, where it is used to initialize or refine the next rounds of local training. In this way, a closed coordination loop is maintained between local learning and global model integration.

From an architectural standpoint, the federated aggregation layer is designed to handle variability in device capabilities, sensing environments, and workload profiles. Because devices can differ markedly in data distributions, compute resources, and uptime, local models are not expected to evolve in the same way. FedAvg offers a systematic way to periodically align these local models without requiring centralized data storage or homogeneous execution patterns across nodes.

Beyond coordinating distributed learning, the aggregation layer also provides system-level coordination signals that reflect the status of the federated process itself. These include, for example, synchronization schedules, update readiness, and global model distribution events. The signals are exposed to other architectural components, as contextual metadata, without imposing assumptions on how they are interpreted or used. Consequently, the federated aggregation layer serves as a coordination and information-sharing backbone, linking local device behavior with network-wide learning activities.

By separating federated coordination from the underlying decision logic and optimization strategies, this layer establishes a modular foundation for distributed intelligence. Within the overall architecture, its role is to determine how learning-related information is disseminated across the IoT network, whereas the concrete use of this information in adaptive offloading or control policies is addressed in later chapters.

3.3 Data Sources and Simulation Environment

This section describes the data sources and simulation setup used to instantiate and assess the proposed IoT system model. It combines real-world sensor readings with synthetically generated data to reflect realistic sensing behavior while preserving full control over the system's dynamics. The simulation environment is based on a discrete-event framework that enables detailed modeling of asynchronous IoT device behaviors, resource consumption, and communication uncertainties.

3.3.1 Real and Synthetic IoT Data

The experimental pipeline starts by combining both real and synthetic sensor readings into a single, unified data format. Real measurements are sourced from the Sutardja Dai Hall (SDH) dataset [73], which contains time-stamped readings from physical sensors deployed in a functioning indoor setting. These data provide a realistic reference point for characterizing sensing behavior in IoT devices.

To supplement physical measurements, additional virtual sensor readings are created using Python-based data generation functions. These synthetic signals are defined within specified ranges to mimic auxiliary sensing channels and system-level indicators that are not directly present in the physical dataset. Merging physical and virtual sensor data supports building a more comprehensive system state while maintaining realism.

All sensor measurements are loaded into a `Pandas` DataFrame to enable structured storage and processing. The timestamps for each reading are converted to standardized date-time objects to maintain temporal consistency throughout the dataset. Precise time synchronization is crucial for time-dependent modeling, especially when simulating battery usage and resource dynamics in a changing IoT environment. This preprocessing step provides a consistent temporal basis for all later modeling and simulation tasks.

3.3.2 Non-IID Data Modeling

To capture the heterogeneity typical of real-world IoT deployments, the simulation data are deliberately modeled as non-identically distributed data (non-IID) across devices. Sensor nodes vary in workload intensity, CPU usage patterns, and energy consumption behavior, leading to various local data distributions. This non-IID property is essential for assessing distributed learning and decision-making methods under realistic conditions.

In addition to supporting non-IID data across devices, the DT framework operates as a virtual representation of the sensor network, continuously reflecting both physical and network-layer characteristics of the underlying IoT system. To maintain consistency between the virtual and physical realms, each DT instance is refreshed either continuously in real time or at regular intervals by synchronizing with its associated sensor node. In this way, the DT can faithfully represent the current operating state of the device, including its energy consumption profile and the progression of its computational workload.

Synthetic training data are generated to emulate battery drain behavior under various operating conditions. Specifically, execution time intervals are sampled uniformly from the range $[0.8, 1.2]$ seconds, while CPU load values are drawn from a constrained interval of $[0.2, 0.8]$ [20]. These limits are selected to approximate realistic device usage scenarios while limiting excessive variability that could destabilize subsequent modeling stages. Using the sampled execution durations and CPU utilization values, a linear battery drain model is trained using `scikit-learn`'s `LinearRegression`. The resulting model captures the relationship between computational intensity, execution time, and energy consumption, and is subsequently used during simulation to estimate incremental battery depletion at each update step. The trained battery drain model is applied in simulation to estimate each sensor node's energy use at every time step. Battery levels are then updated incrementally according to the predicted drain, allowing continuous monitoring of energy depletion over time. Because individual nodes face different CPU workloads and task arrival patterns, their battery trajectories naturally diverge, further emphasizing the system's non-IID characteristics.

By continuously updating the DT state variables with both observed and predicted battery behavior, the simulation maintains a coherent virtual representation of energy flows across heterogeneous sensor nodes. This tight integration ensures that energy consumption patterns evolve plausibly over time and differ naturally between devices, thereby reinforcing the non-IID characteristics of the simulated IoT environment.

In addition to energy modeling, short-term CPU usage is estimated using a Simple Moving Average (SMA). A deque maintains a sliding window of recent CPU load samples, over which the mean is calculated to obtain a smoothed view of current processing demand. This method offers lightweight predictive insight into CPU behavior without the need for sophisticated time-series forecasting techniques.

Wireless network latency is represented as an independent stochastic process. Individual latency values are sampled from a Gaussian distribution [74], reflecting the inherent randomness of wireless communication caused by channel fluctuations and background traffic. This stochastic approach introduces further variability among devices, since the measured latency changes over time and between nodes.

3.3.3 Simulation Framework (SimPy-based)

The overall system behavior is modeled using a discrete-event simulation framework built with SimPy. Each IoT sensor node is represented as its own SimPy process, operating as an autonomous device that operates asynchronously within the network. This approach enables the simulation to reflect concurrent execution, event-driven communication, and diverse device behaviors.

At every simulation step, a sensor node updates its internal state through a series of actions, such as sampling and averaging CPU load, calculating battery usage, and generating network latency values.

This state is then used to adjust both the behavior of the local device and to produce historical logs for predictive analysis. Specifically, past battery drain patterns under different CPU loads are examined by the DT component to estimate short-term energy consumption. In the same way, recorded latency values are reviewed to detect possible congestion trends. These predictive indicators are then sent to local

decision interfaces, allowing devices to factor in forward-looking information when choosing where to execute their tasks.

By combining real sensor data, controlled synthetic inputs, and a discrete-event simulation engine, the proposed framework offers a flexible and reproducible environment to evaluate adaptive task execution in IoT systems. The SimPy-based design enables detailed modeling of resource dynamics, asynchronous interactions, and stochastic variability, offering a realistic testbed for the predictive and adaptive mechanisms presented in subsequent chapters. Building on the system modeling formulation, the next chapter lays out the theoretical foundations of the proposed framework. It presents the formal principles of the key technologies and decision processes, as well as the mathematical structures regulating distributed optimization in the presence of uncertainty.

Chapter 4

Theoretical Formulations of Federated Multi-Agent Reinforcement Learning

4.1 Reinforcement Learning Foundations

RL is a type of ML task focused on sequential decision-making, where an agent interacts with an environment and gradually learns a policy that maximizes long-term cumulative rewards. In contrast to supervised learning, RL depends on the evaluative feedback gathered through this interaction rather than the labeled examples. This makes RL especially well-suited for dynamic and uncertain settings, such as resource-constrained IoT environments. A detailed and formal treatment of the RL theory can be found in [75, 76].

An RL problem is commonly modeled as a Markov Decision Process (MDP), defined by the tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle, \tag{4.1}$$

where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P(s' \mid s, a)$ the probability of state transition, $R(s, a)$ the immediate reward function and $\gamma \in [0, 1)$ the discount factor that controls the influence of future rewards [76].

At each discrete time step t, the agent observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to a policy $\pi(a \mid s)$, and receives a reward $r_t = R(s_t, a_t)$. The

environment then transitions to a new state s_{t+1} following the transition dynamics P. The agent's objective is to learn an optimal policy π^* that maximizes the expected discounted return

$$G_t = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right], \tag{4.2}$$

as defined in Equation (4.2) [75].

To evaluate policies, RL introduces value functions that quantify the expected long-term reward. The state-value function in a policy π is defined as

$$V^{\pi}(s) = E_{\pi}\left[G_t \mid s_t = s\right], \tag{4.3}$$

while the action-value function is given by

$$Q^{\pi}(s,a) = E_{\pi}\left[G_t \mid s_t = s, a_t = a\right]. \tag{4.4}$$

These value functions satisfy the recursive Bellman expectation equations, which relate immediate rewards to expected future values [77].

Optimal behavior is characterized through the optimal value functions $V^*(s)$ and $Q^*(s,a)$, which satisfy the Bellman optimality condition. In particular, the optimal action-value function fulfills

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) \max_{a' \in \mathcal{A}} Q^*(s',a'), \tag{4.5}$$

as shown in Equation (4.5). Many RL algorithms aim to approximate this function through iterative updates based on observed state transitions [78, 79].

4.2 Multi-Objective Formulation

Our architecture addresses two key challenges in distributed IoT sensor networks: preserving energy efficiency and achieving strong learning performance. Aggressive offloading can reduce local CPU load but may incur high energy costs, while performing all computation locally can save offloading energy at the expense of learning quality. We therefore formulate the problem as a multi-objective optimization task

to explicitly model these trade-offs. The total energy consumption, defined in Equation (4.6), is computed as the sum of the energy used for local processing and for offloading across all sensor nodes:

$$E_{\text{total}} = \sum_{i=1}^{N} \left(E_{\text{local},i} + E_{\text{offload},i} \right), \tag{4.6}$$

where $E_{\text{local},i}$ is the energy consumed by node i for local processing, and $E_{\text{offload},i}$ represents the additional energy cost incurred when offloading tasks.

We formulate a multi-objective optimization problem designed to promote high battery levels in the system and permit occasional offloading, while balancing energy usage and learning performance across the sensor network. Learning performance is captured by a normalized metric L_{perf}, such as model accuracy, convergence rate, or loss reduction, with $L_{\text{perf}} \in [0, 1]$ and 1 corresponding to optimal performance. Following the weighted-sum scalarization approach commonly adopted in multi-objective optimization [80], these objectives are combined into the unified cost function defined in Equation (4.7):

$$\min_{x} \quad \lambda\, E_{\text{total}} + (1 - \lambda)\,(1 - L_{\text{perf}}) - \beta\, R(b), \tag{4.7}$$

subject to the restriction:

$$\text{latency}(i) \leq \ell_{\max}, \quad \forall\, i \in \{1, \ldots, N\}, \tag{4.8}$$

where:

- x represents the variables that influence offloading choices, including whether to offload to an edge server or process locally,
- $\lambda \in [0, 1]$ is a weighting factor that finds a balance among learning performance and consumption of energy.
- $\beta > 0$ scales the battery bonus,

- $R(b)$ is a reward bonus function defined as

$$R(b) = \begin{cases} 1, & \text{if } b > 85\%, \\ 0, & \text{otherwise,} \end{cases} \tag{4.9}$$

- $\ell_{\max}$ is the maximum allowable latency.

When λ is near 1, the model emphasizes minimizing energy, whereas values near 0 place greater importance on learning performance.

In our framework, the agent's reward function is constructed to mirror the negative of the objective value presented in Equation (4.7). The reward is computed by applying penalties for high latency and for energy usage, where the latter depends on local CPU utilization and offloading costs, and by adding an extra penalty whenever the battery level falls below the critical threshold of 30%. In contrast, a bonus is granted when the battery level is maintained above 85% [20]. Thus, by incorporating the bonus term $R(b)$, our approach promotes policies that achieve this target behavior rather than enforcing a strict constraint $b \geq 85\%$. This formalization provides the theoretical basis for our method and justifies the construction of the RL reward function. As a result, the learned policy simultaneously maintains the battery level above 85%, achieves strong learning performance, and performs offloading conservatively.

The goal of our RL agent is to learn the response to the multi-objective problem. Each sensor node in the system observes a status vector composed of network latency, CPU utilization, and battery charge level. Based on this state, the agent selects an action $x \in \{\text{local}, \text{offload}\}$. The reward function is defined as:

$$\begin{aligned} \text{reward} = -\Big(&\text{energy\_used} + \lambda_{\text{lat}} \cdot \text{latency} \\ &+ \text{battery penalty} + \text{extra offload penalty}\Big) + \text{bonus}(b). \end{aligned} \tag{4.10}$$

where:

- energy_used indicates the amount of battery used as a result of offloading or

local computation.

- λ_{lat} is the delay penalty's scaling factor,
- the *battery penalty* is used when the battery level drops below a key threshold which is 30%, therefore discouraging states with very low energy.
- the *extra offload penalty* it occurs throughout the offloading process, representing the increased cost,
- bonus(b) encourages the agent to keep large energy reserves by offering a positive reward if the battery level b surpasses 85%.

This reward function acts as a *soft constraint*, assigning penalties to actions that cause high energy use and rewards to those that keep the battery level above 85%. Consequently, the RL agent learns to trade off offloading and local computation, choosing to offload only when it lowers local CPU usage without depleting overall energy reserves.

Therefore, the negative of our multi-objective cost function is given by:

$$\min_{x} \quad \lambda\, E_{\text{total}} + (1-\lambda)\,(1 - L_{\text{perf}})\,. \tag{4.11}$$

Equation (4.11) is represented in practice by our RL reward. This formulation, together with the delay constraint, provides a solid theoretical basis for our design choices. Moreover, it allows the integration of additional constraints, such as network capacity, and ensures that the sensor network attains advantageous learning performance while making proactive and energy-efficient offloading decisions.

4.3 Multi-Agent Reinforcement Learning (MARL)

MARL extends classical RL to settings where multiple autonomous agents act at the same time within a common environment and interact with each other [81, 82]. Unlike single-agent scenarios, the view of each agent of the environment and the results of its actions depend on the behavior of other agents, making the environment

non-stationary from the point of view of that agent. This property significantly complicates learning, stability, and convergence [83, 84].

4.3.1 Markov Games and MARL Formulation

The theoretical basis of MARL is typically formulated in terms of a *Markov Game* [85], also called a stochastic game. A Markov Game is specified by the tuple:

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, P, \{R_i\}_{i\in\mathcal{N}}, \gamma \rangle,$$

where $\mathcal{N}$ denotes the set of agents, $\mathcal{S}$ the global state space, $\mathcal{A}_i$ the action space of agent i, $P(s' \mid s, a_1, \ldots, a_{|\mathcal{N}|})$ the state transition probability, R_i the reward function of agent i, and $\gamma \in [0, 1)$ the discount factor.

Each agent aims to learn a policy $\pi_i(a_i \mid s)$ that maximizes its expected discounted return,

$$J_i = E\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^1, \ldots, a_t^{|\mathcal{N}|})\right],$$

while taking into consideration the changing policies of other agents. In cooperative scenarios, agents pursue common goals, while competitive or mixed scenarios feature opposing or partially conflicting rewards.

4.3.2 Prioritized Experience Replay

To improve sample efficiency and accelerate convergence, we employ the proportional PER mechanism proposed by Schaul et al. [86]. In this approach, a prioritized replay buffer samples transitions with larger temporal-difference (TD) errors more frequently. In concrete terms, each stored transition k is assigned a priority:

$$p_k = \left|\delta_k\right|^{\alpha}, \quad \delta_k = y_k - Q\big(s_k, a_k\big), \quad \alpha = 0.7. \tag{4.12}$$

In Equation (4.12), y_k is the target Q-value (Equation (4.16)) and $Q(s_k, a_k)$ its current estimate. Sampling probability for transition k is then

$$P(k) \;=\; \frac{p_k}{\sum_j p_j}, \tag{4.13}$$

Equation (4.13) ensures that experiences with larger errors are sampled more frequently.

To compensate for the bias caused by this non-uniform sampling, each transition is assigned a weight defined as follows:

$$w_k \;=\; \Big(|\mathcal{B}|\, P(k)\Big)^{-\beta}, \quad \beta = 0.5, \tag{4.14}$$

In Equation (4.14), $|\mathcal{B}|$ denotes the current buffer size. Importance-sampling weights are applied during gradient updates to preserve an unbiased estimate of the loss gradient. By tuning α and β, the method balances prioritizing specific transitions with adhering to theoretical convergence guarantees, thereby improving both the stability and the speed of learning.

4.3.3 Dueling Double Deep Q-Network (D3QN)

We employ a D3QN [87] to approximate the optimal state–action value function in resource-constrained IoT environments. The dueling network architecture proposed by Wang et al. [88] decomposes the Q-value into a state-value term $V(s;\theta)$ and an advantage term $A(s,a;\theta)$, which are then combined as

$$Q(s,a;\theta) = V(s;\theta) + \Big(A(s,a;\theta) - \tfrac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta)\Big), \tag{4.15}$$

Equation (4.15) has been demonstrated to speed up convergence in settings where many actions produce comparable returns.

To mitigate the overestimation bias commonly observed in single-network Q-learning, we adopt the Double-DQN target update scheme proposed by van Hasselt et al. [89]:

$$y \;=\; r + \gamma\, Q\Big(s', \arg\max_{a'} Q(s',a';\theta)\;;\;\theta^{-}\Big), \quad \gamma = 0.95, \tag{4.16}$$

In Equation (4.16), θ^{-} denotes the parameters of a target network that are kept fixed and updated to match the online network every three training episodes.

4.4 Federated Learning Process

Within the framework, FL is used to support collaborative model training while reducing centralized communication costs and maintaining data on the devices where it is generated. Instead of sending raw sensor readings to a central server, each IoT node trains its own local predictive model on its private dataset, which can be enriched with outputs from virtual sensors. This setup mirrors realistic IoT environments, where devices are heterogeneous in terms of computation power, battery constraints, and data availability. As a result, the local learning models can vary from simple linear regressors to compact neural networks, depending on each device's processing capabilities.

During each FL round, local training is carried out for a predefined number of epochs. Once this local optimization step is finished, devices send only their updated model parameters to a central aggregation node, without exposing any raw data.

4.4.1 Federated Averaging Mechanism

FL enables collaborative model training across distributed devices without centralizing raw data. In the canonical FedAvg framework introduced by McMahan et al. [90], each participating device performs multiple local stochastic gradient descent (SGD) steps on its private dataset before transmitting the updated model parameters to the central server.

Let $\theta_i^{(t)}$ denote the model parameters on device i after its local training in communication round t, and let n_i be the number of local training samples stored on that device. The server then combines the local updates via a weighted averaging scheme given by

$$\theta^{(t+1)} = \frac{\sum_{i=1}^{N} n_i \, \theta_i^{(t)}}{\sum_{i=1}^{N} n_i}, \tag{4.17}$$

In (4.17) N the number of devices participating in the current round is indicated. The weighting mechanism ensures that devices with larger local datasets exert a proportionally greater influence on the updated global model. After aggregation, the refined global parameters $\theta^{(t+1)}$ are broadcast to all participating devices, thus

initiating the subsequent training round.

In the proposed system, aggregation and model distribution are performed synchronously, so a communication round finishes only once all chosen devices have provided their updates.

This approach guarantees that devices holding larger local datasets have a stronger impact on the resulting global model. After aggregation, the refined global model is distributed back to all participating devices and is used as the starting point for the next FL round. In the system under study, both aggregation and model distribution occur synchronously, implying that each round advances only once all involved devices have transmitted their updates.

Beyond its role in distributed learning, the FL process closely interacts with system-wide resource management. Performing local training generates computational and energy overhead, which is shaped by adaptive offloading strategies. By continuously deciding whether tasks are executed on the device or shifted to edge resources, the system indirectly controls the computational burden linked to local model updates. The integration of virtual sensor data further enhances local datasets, enhancing model robustness and prediction performance in the presence of heterogeneous and non-IID data. Through iterative FedAvg, the framework exploits the shared intelligence of the entire network to build a resilient, adaptive global model while preserving decentralization and resource-aware learning behavior.

4.4.2 Federated Learning for MARL

To utilize the aggregated information from all sensor nodes without collecting their raw measurements in a central location, we embed a synchronous FL cycle into the MARL control loop. At each control step, every node i refines its local model implemented either as a linear regressor or a compact multilayer perceptron by running E epochs of gradient descent on its augmented dataset. The parameter E may be tuned according to the available CPU capacity, however, in our implementation it is kept fixed for simplicity.

After completing local training, each node sends only its model parameters θ_i along with the corresponding number of samples n_i. The central server then com-

putes the weighted average of these updates:

$$\theta_{\text{global}} \;=\; \frac{\sum_{i=1}^{N} n_i\,\theta_i}{\sum_{i=1}^{N} n_i}, \tag{4.18}$$

Equation (4.18) ensures that clients with larger datasets exert a proportionally greater influence on the global model. After aggregation, the updated global parameters θ_{global} are then broadcast to all clients, where they overwrite the local weights before the next decision step. The FedAvg approach keeps training communication-efficient by exchanging only parameter vectors, and preserves privacy since the raw data never leave the edge devices.

4.5 Explainability Methods

To make the agents' decision process more interpretable, we compute a gradient-based saliency map for each selected action following the approach introduced by Simonyan et al. [91]. Given a state s and the chosen action $a^\star = \arg\max_a Q(s, a; \theta)$, we define the corresponding saliency vector as

$$g \;=\; \Big|\nabla_s\, Q\Big(s, a^\star; \theta\Big)\Big|. \tag{4.19}$$

In Equation (4.19), the absolute gradient magnitude reflects how strongly each state feature contributes to the action's value. These feature-level attributions are normalized and logged at every decision step, enabling post-hoc examination of the agent's emphasis on battery level, CPU load, latency, and historical CPU usage patterns. While this approach yields interpretable information about the decision drivers at each timestep, it also has limitations. First, computing gradients during inference introduces additional overhead, which may affect latency-critical IoT scenarios. Second, the resulting saliency map is static and lacks temporal abstraction, so it cannot represent dependencies across successive states or capture evolving trends. The following chapter describes the proposed framework in detail. It defines the system architecture and combines the previously introduced components into a unified solution designed for resource-limited IoT environments.

Chapter 5

Proposed Framework

5.1 Framework Architecture

This chapter introduces the proposed framework for predictive, resource-aware computation offloading and distributed learning in heterogeneous IoT networks. The framework is implemented in Python in a Jupyter Notebook environment and uses the SimPy library [92] to emulate the asynchronous behavior of IoT devices via discrete-event simulation. Each sensor node is modeled as an independent SimPy process with its own event timeline, allowing the simulation to reflect decentralized operation, fluctuating workloads, and time-varying wireless conditions typical of real-world deployments.

5.1.1 Layered Design and Component Roles

The framework follows a layered design to separate sensing and local execution from prediction, learning coordination, and decision making. The main layers are summarized as follows.

- **Physical and Virtual Sensing Layer:** Each IoT node produces tasks and measurements based on physical sensor streams (SDH dataset [93]) and additional virtual sensor values generated within specified ranges.
- **DT Prediction Layer:** A lightweight DT keeps an up-to-date virtual model of each node and generates near-term predictions of resource usage and net-

work behavior (e.g., anticipated battery depletion, CPU load patterns, and latency changes).

- **LLM-Guided Decision Layer:** A prompt engine builds structured natural-language prompts by combining (i) DT predictions, (ii) the current node status and queue information, and (iii) optional FL convergence signals. An LLM then uses in-context reasoning on these prompts to produce an offloading decision (local vs. edge execution), optionally adding a short explanatory signal.
- **Execution and Control Layer:** Each node carries out the chosen action by running tasks locally or offloading them to the edge, then revises its local state (battery level, CPU utilization, queue) according to the actual computation and communication costs, thereby closing the control loop for the subsequent decision epoch.
- **FL Coordination Layer:** Nodes independently train local models and, at intervals, send their model updates to a central federated aggregator, which synchronously combines them (e.g., via FedAvg) and then broadcasts the refreshed global model back to the nodes for the next training rounds.

This separation supports modular evaluation: prediction quality (DT), learning coordination (FL), and task execution decisions (offloading) can be analyzed both independently and as a closed-loop system.

5.1.2 Information and Control Flow

At runtime, system components interact through a coordinated exchange of prediction, learning, and control information, which supports adaptive and proactive offloading decisions in dynamic IoT environments. Instead of concentrating on the internal models of each component, the focus in this part is on how data, predictions, and control signals move through the system during execution. This control flow enables proactive, context-aware decision-making in diverse and changing IoT conditions.

Figure 5.1 shows the interaction flow among the sensor nodes, their corresponding

DTs, the LLM-based decision controller, the offline RL module, and the FedAvg stage.

State Reporting and Synchronization. At every decision step, each IoT sensor node gathers its current operating status, such as remaining battery charge, real-time CPU utilization, measured wireless communication delay, and the state of its task queue. This collected status data is then sent to its associated DT, which preserves an up-to-date virtual model of the physical node. Through this continuous update process, the DT is kept closely aligned with the node's real-world condition before any control or management action is executed.

Predictive Feedback from the Digital Twin. Upon receiving the updated state, the DT generates short-term predictions for key resource and network indicators, including projected battery consumption, upcoming CPU load, and likely latency evolution over the next interval. These prediction results are sent back to the sensor node while also being made available to higher-level control and management logic. By using forecasted, forward-looking data instead of only real-time measurements, the system enables anticipatory, proactive offloading decisions that are better aligned with upcoming conditions.

Context Construction for Decision Making. The sensor node transmits its current operational state, along with the DT prediction output, to a dedicated prompt construction module. At the same time, the FedAvg layer provides global learning metrics, such as indicators of convergence progress and update stability while still preserving data privacy by not exposing any underlying raw samples. In addition, a curated subset of high-reward historical offloading examples, derived from offline RL policies, is provided as a reference. The prompt construction module integrates all of these inputs and composes them into a well-structured natural-language prompt that captures the local runtime conditions, anticipated future behavior, and the overall status of the global learning process in a coherent summary.

LLM-Based Offloading Control. The composed prompt is passed to the LLM-based decision controller, which, via in-context learning, determines an offloading

action. Its output indicates whether the ongoing task should run on the local device or be delegated to an edge server. Crucially, this decision process does not rely on any form of online policy training; rather, the LLM directly reasons over the supplied contextual signals and system state. This design enables the controller to quickly adjust its decisions as system conditions evolve, such as fluctuations in resource availability, network latency, or workload characteristics.

Execution and Feedback Loop. The sensor node promptly carries out the received decision, adjusting its task queue and resource usage as needed. The results of execution such as task completion latency and energy usage, are captured in the node's subsequent state updates. These state updates complete the control loop by supplying fresh observations to the DT and the federated learning process, supporting continuous system adaptation.

Federated Learning Coordination. In parallel with executing their assigned tasks, the sensor nodes periodically carry out local training of their models and then send encrypted model updates to the federated aggregator. The aggregator performs synchronous aggregation over all received updates to compute a new global model and then distributes the refreshed parameters back to the participating nodes. Although the specific learning algorithms and optimization details are presented in a different section, the resulting convergence metrics such as loss trends or accuracy improvements, are subsequently fed back into the control loop, where they serve as contextual signals that guide decisions in the next decision epochs.

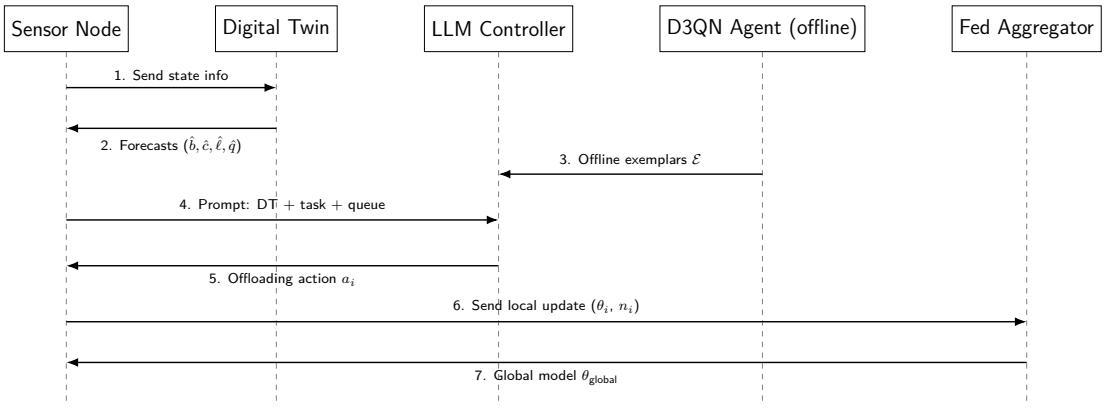


Figure 5.1: System interaction sequence diagram.

5.1.3 Simulation Backbone (SimPy Process Model)

The simulation environment is organized around asynchronous device processes. Each IoT device is instantiated as a SimPy process that repeatedly executes the following operations: (i) task arrival and queue update, (ii) device state update (CPU load, battery level), (iii) network latency sampling, and (iv) decision execution (local vs. offload). This process-level abstraction captures device heterogeneity and decentralized timing, enabling realistic evaluation under varying loads and wireless conditions.

5.1.4 Data Preprocessing and Battery Modeling

Sensor data are imported into a `pandas` DataFrame, including timestamps and physical measurements from the SDH dataset [93], together with virtual sensor values generated by a Python code within predetermined ranges. Timestamps are converted to date-time objects to ensure temporal alignment across all streams, which is essential for time-dependent modeling (e.g., battery evolution and workload dynamics).

To emulate energy depletion under computation, synthetic training samples are generated for a battery drain model. Time intervals are drawn uniformly from $[0.8, 1.2]$ seconds and CPU load values are sampled from $[0.2, 0.8]$, reflecting moderate operating regimes while avoiding excessive variability that could destabilize downstream estimates. A linear battery drain model is then trained using `scikit-learn`'s `LinearRegression` and is used during simulation to update each node's battery level at every step based on its current load and execution duration.

5.1.5 Network Latency Simulation

Wireless latency is modeled as a stochastic process [94] to reflect variability in channel conditions and background traffic. At each step, the baseline latency is sampled from a Gaussian distribution with mean 30 ms and standard deviation 3 ms using `random.gauss(30, 3)`. To emulate sporadic congestion, an additional delay is injected with probability 5%; when triggered, the extra delay is sampled uniformly

from $[5, 10]$ ms via `random.uniform(5,10)` [20]. The resulting latency stream captures both typical operation and occasional congestion spikes, allowing offloading decisions to be evaluated under dynamic network conditions.

Each IoT node is associated with a lightweight DT that maintains a continuously updated virtual representation of its operational state. At each simulation step, nodes provide the DT with key telemetry, including current battery level, instantaneous CPU load, and observed wireless latency. Based on these signals, the DT produces short-horizon forecasts that are computationally inexpensive yet informative for proactive control.

In particular, the DT estimates expected battery drain using the trained linear regression battery model, conditioned on the node's current load and the upcoming execution interval. In parallel, it computes a smoothed estimate of CPU demand using a moving average over recent load samples stored in a deque. These predicted quantities form near-term indicators of resource stress (e.g., impending battery drop or sustained high CPU), enabling the control layer to reason beyond instantaneous measurements. The DT then returns these forecasts to the node (and to the prompt engine), establishing a bidirectional interaction that supports proactive offloading.

5.1.6 Simulation Configuration

All experiments are conducted within a controlled simulation environment, allowing strict regulation of evaluation conditions and ensuring reproducibility of results. Unless otherwise specified, each experiment consists of 350 independent episodes, with every episode unfolding over 35 discrete time steps. At each time step, all virtual IoT devices independently update their operational state and perform task scheduling decisions. This design emulates the asynchronous and distributed nature of real-world IoT systems, where devices operate autonomously while interacting within a shared environment.

In each episode, the simulated network comprises 5 heterogeneous sensor nodes, representing resource-constrained IoT devices with varying battery reserves, computational loads, and network conditions. This configuration strikes a balance between realism and computational efficiency while preserving inter-device interaction and

overall system dynamics. The number of nodes remains fixed across all experiments to ensure that observed performance variations stem from differences in decision-making strategies and predictive mechanisms rather than scalability effects.

Upon completion of every episode, a set of performance indicators is recorded for evaluation and comparison. These metrics include the mean residual battery level across all devices, the total number of offloading operations performed during the episode, and controller-specific statistics such as the distribution of selected actions and the temporal consistency of decisions. Collectively, these measures provide a comprehensive view of energy consumption behavior, workload allocation patterns, and the adaptive stability of the framework under time-varying and uncertain operating conditions.

5.2 Digital Twin Modeling

5.2.1 Digital Twin Modeling and Predictive Intelligence

In the proposed framework, each IoT device is associated with a lightweight, data-driven DT that maintains a continuously updated virtual representation of the device's operational state. Unlike high-fidelity or physics-based DT, which are computationally intensive and unsuitable for large-scale IoT deployments, the DT adopted in this thesis is intentionally designed as a compact ML model. This design choice ensures low computational overhead, fast update cycles, and practical deployability in resource-constrained IoT environments.

The DT stays aligned with its physical device by periodically gathering runtime telemetry such as battery status, instantaneous and average CPU load, measured wireless latency, local task queue length, and selected learning-related metrics. These data are combined and normalized into a compact state description that reflects both the device's current operating state and its recent activity. By transforming raw sensor and system readings into organized state variables, the DT offers a stable, interpretable layer between low-level measurements and higher-level decision processes.

At the core of the DT is a lightweight predictive modeling unit based on lin-

ear regression. Synthetic and historical operational data are leveraged to train regression models that forecast short-term resource behavior, in particular battery consumption as a function of CPU load and execution duration. Linear regression is intentionally chosen for its low computational cost, transparency, and reliability when only limited data are available. This enables each DT instance to be trained quickly and updated incrementally, without incurring substantial computation or memory overhead on the device or at the edge.

Beyond battery prediction, the DT also keeps short-term estimates of CPU load trends via SMA, and it represents wireless latency as a stochastic process using statistically generated samples. Together, these elements create an integrated ML-based DT that reflects key temporal behaviors of the device without relying on complex deep learning or simulation-intensive models. This yields a predictive model that is expressive enough to enable proactive decisions, yet lightweight enough to be deployed across large-scale IoT networks.

The DT's predictive outputs allow the system to foresee short-term resource limitations, including impending battery drain, CPU overload, or rising network latency. These predictions are shared with both the local control logic and the centralized LLM-based offloading module. Instead of responding only after performance has degraded, the system uses DT forecasts to reason about future conditions and proactively adapt its task execution strategies.

To integrate smoothly with the LLM controller, DT predictions are represented as structured, semantically rich summaries that capture both current observations and near-term forecasts. By conveying predictive data in this compact, interpretable format, the DT enables the LLM to decide on offloading actions based on projected system dynamics, without the need for explicit retraining or intricate policy optimization.

Overall, within this framework, the DT acts as a *lightweight, ML-based prediction layer* that improves situational awareness and enables foresight-driven computation offloading. Using linear regression and basic statistical estimators, the DT strikes a practical compromise between predictive accuracy, interpretability, and computational cost. This design is essential for achieving scalable, energy-efficient, and

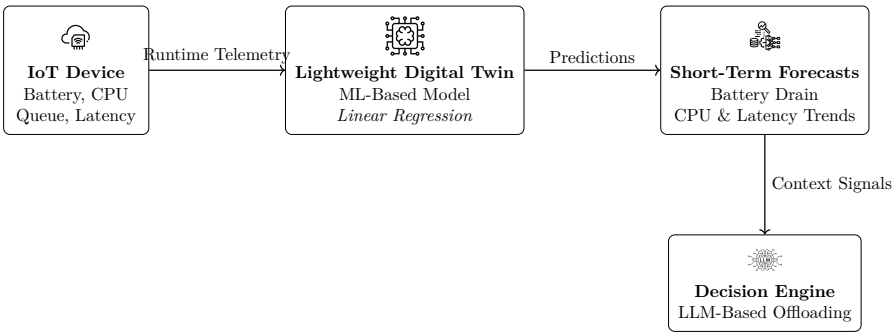


Figure 5.2: Digital Twin for proactive offloading.

proactive task offloading in federated IoT systems, where resource limitations and rapidly changing conditions are commonplace [95].

5.3 Federated Multi-Agent Reinforcement Learning

The framework integrates Federated MARL as a distributed learning backend that enables adaptive computation offloading while keeping data local. Each IoT node acts as an independent learning agent, training locally on its own data streams, which include both physical sensor measurements and virtual sensor features produced in the simulation environment. Raw data remain on the device at all times, protecting privacy and lowering communication costs.

Local training is executed independently on each node for a fixed number of epochs in every federated round. After finishing the local training, each node sends only its updated model parameters to a federated aggregation server located at the edge. This server performs synchronous model aggregation using the FedAvg algorithm and then distributes the resulting global model to all participating nodes. The global model initializes the subsequent local training phase, maintaining consistent learning dynamics across heterogeneous devices with non-IID data.

In the proposed system, the FL loop goes beyond simply improving the model accuracy. It also generates runtime learning metrics that characterize the overall training state, including convergence progress, loss trends, and the stability of updates across nodes. These metrics are collected in each round of aggregation and are

provided as structured signals to the decision-support pipeline. This makes learning dynamics visible as system variables instead of remaining hidden background processes.

Federated MARL does not make offloading decisions directly during the execution of online tasks. Rather, it runs offline and asynchronously relative to the execution of tasks. During training, federated MARL identifies high-reward state–action trajectories, stores them as exemplar cases, and then leverages these exemplars as references at runtime when decisions must be made. Furthermore, FL convergence indicators are fed into the prompt generation module so that the online decision-making logic can explicitly consider the current training and convergence status of the distributed network when selecting actions.

Algorithm 1 specifies the offline federated MARL procedure employed to obtain shared representation models and prototypical decision exemplars. The process advances in synchronized federated rounds, in which each IoT node carries out local MARL training for a predetermined number of epochs and transmits only its model parameters to the central aggregation server.

The aggregator uses FedAvg weighted to build a global model, maintaining consistent parameter alignment across various nodes. During model updates, each node records high-reward state–action transitions observed during local training. These transitions are then filtered and stored as compact exemplars that capture effective offloading strategies under varying operating conditions.

The resulting global model parameters and exemplar set are exported as static artifacts rather than deployed as online control policies. They function as reference knowledge for downstream decision components, allowing lightweight integration with predictive and reasoning-based modules without incurring extra reinforcement learning training overhead.

5.4 Explainability Layer

To enable post-hoc interpretability of offloading decisions, the framework uses gradient-based saliency analysis on the D3QN's learned value function. In the dueling architecture, the action-value is split into a state-value stream and an advantage stream,

Algorithm 1 Federated MARL Training

Require: Set of IoT devices $\mathcal{N}$, local datasets $\{\mathcal{D}_i\}$, number of FL rounds R
Ensure: Global model $\mathbf{w}$, offline exemplar set $\mathcal{E}$

```
Initialize global model w^(0)
for r = 1 to R do
    for all devices i ∈ N in parallel do
        Train local MARL model on D_i for E epochs
        Extract high-reward state–action samples
        Send local model parameters w_i^(r) to aggregator
    end for
    Aggregate updates via FedAvg to obtain w^(r)
    Broadcast w^(r) to all devices
    Record convergence indicators (loss, stability)
end for
Store selected high-reward samples in exemplar set E
Expose E and FL convergence signals to runtime prompt engine
```

allowing the model to independently represent the overall quality of a system state and the relative effect of different actions. Saliency analysis is then applied to the final action-value corresponding to the chosen action, providing feature-level attribution without altering the network architecture.

At each decision step, the gradients of the chosen action-value are computed considering the input state features. The absolute values of these gradients capture how sensitive the action valuation is to small changes in individual state variables. In particular, saliency is obtained from the local sensitivity of the action-value output to the normalized state input, giving a first-order estimate of how much each characteristic influences the decision at that step. These saliency values are then normalized and recorded for later offline analysis, offering a quantitative explanation of which system factors had the greatest impact on a specific offload decision.

In the considered IoT scenario, saliency scores make it possible to pinpoint the main factors driving decisions between local execution and offloading for example, low battery levels, high CPU load, or increased network latency. Because the dueling architecture explicitly distinguishes overall state value from action-specific advantages, the resulting saliency profiles remain more consistent in states where several actions have comparable expected returns, thereby enhancing interpretability in ambiguous situations.

From an implementation perspective, saliency is computed only during offline

analysis and never during live control. While this requires extra computation for gradient evaluation, the added cost is minor compared to retraining the policy or using separate explainability models. Nonetheless, the method is fundamentally local and instantaneous: it indicates feature relevance at a single decision point and does not account for temporal relationships or delayed effects along state trajectories.

Despite these constraints, gradient-based saliency remains a practical and lightweight explainability tool for D3QN-based offloading policies, allowing systematic inspection of learned behavior and enhancing transparency in autonomous resource management decisions.

Table 5.1 presents the semantic meaning of high saliency values for each state feature considered in the gradient-based explainability analysis. By linking individual input dimensions to their functional role, the table clarifies how the decision model ranks different system aspects when choosing offloading actions. High saliency values imply that small changes in the associated feature substantially affect the estimated action value, indicating whether decisions are mainly influenced by energy limitations, processing demand, communication quality, or short-term workload dynamics. This feature-level perspective facilitates post-hoc evaluation of the controller's behavior and helps confirm that the learned policies remain consistent with the intended, domain-specific design goals.

Table 5.1: State features and their interpretation in gradient-based saliency analysis.

State Feature	**Interpretation of High Saliency**
Battery level (b)	Decision is primarily driven by energy preservation; high influence typically correlates with offloading under low battery conditions.
Instantaneous CPU load (c)	Indicates computational saturation, high saliency suggests avoidance of local execution due to processing bottlenecks.
Network latency (ℓ)	Reflects communication constraints, high saliency implies sensitivity to delay, often discouraging offloading under congestion.
Average CPU load ($\bar{c}$)	Captures short-term workload trends, elevated saliency indicates anticipation of sustained computation pressure rather than transient spikes.

5.5 LLM-Assisted Decision Intelligence

Adaptive computation offloading in resource-limited IoT systems is fundamentally a sequential decision problem under uncertainty. At each decision point, a node must decide between processing tasks locally or offloading them to the edge, while considering time-varying energy levels, computational demand, and network state. In the proposed framework, this decision-making is realized as a closed-loop control scheme that closely couples predictive models, collaborative learning feedback, and execution management.

At runtime, the decision controller is provided with a structured snapshot of the system context, which includes: (i) the node's current operating state, (ii) short-term predictions produced by the DT, such as estimated battery depletion and smoothed CPU utilization, (iii) measured or forecast wireless latency, and (iv) optional learning metrics from the FL procedure, such as convergence stability or loss evolution. Using this contextual information, the controller issues a binary decision (`local` or `offload`), which is immediately applied by the node.

This design explicitly integrates prediction (DT), learning coordination (FL), and execution control (offloading) into a single decision loop. In contrast to traditional offloading schemes that depend on instantaneous measurements or fixed pre-trained policies, the proposed method supports proactive, context-aware choices that forecast imminent resource limitations instead of merely reacting once performance has already deteriorated.

LLM-Driven Offloading Decision-Making

Conventional RL and MARL methods are widely used for computation offloading in IoT systems. Although they work well in stationary or moderately dynamic scenarios, they encounter practical challenges in large-scale, heterogeneous environments. In particular, they demand extensive environment interaction for training, careful hyperparameter tuning, and repeated retraining as system conditions evolve. These factors incur significant computational overhead and reduce responsiveness under rapidly changing workloads and network conditions. To overcome these constraints, the proposed framework replaces online policy learning with an inference-driven decision mechanism based on a LLM. Rather than learning an explicit control policy through exploration, the LLM infers offloading actions directly from structured contextual inputs using in-context reasoning. This shifts the decision-making paradigm from training-intensive optimization to prompt-driven inference.

At each decision point, DT modules linked to individual nodes produce short-term forecasts of key system metrics, such as projected battery level, CPU load, queue dynamics, and wireless delay. These predictions are fused with the node's current state and global learning signals derived from the federated aggregation step. The combined information is then encoded into a structured prompt that captures both the current operating state and the anticipated near-future system behavior.

To further stabilize and steer decision-making, the prompt can include a small set of representative offloading examples obtained from offline-trained MARL policies. These examples encode high-reward decision patterns observed under comparable system conditions and act as reference points that ground the LLM's reasoning.

Notably, this transfer of knowledge is achieved without any need for online policy execution or gradient-based updates.

Once the prompt is formed, it is sent to a pretrained LLM, which produces an offloading decision via semantic reasoning over the given context. Since inference occurs in a single forward pass, the decision latency is bounded and does not depend on the size of the environment or the dimensionality of the action space. Consequently, this method is well suited for use in latency-critical IoT deployments. From an engineering standpoint, the LLM-based controller offers several benefits compared with conventional MARL agents. First, it removes the need for online training and retraining, substantially lowering computational overhead. Second, it generalizes more robustly to unseen operating conditions by reasoning over contextual descriptions instead of relying on fixed state–action mappings. Third, incorporating DT forecasts supports proactive decisions that anticipate upcoming resource pressure, enhancing both energy efficiency and quality of service. Finally, FL indicators supply a form of global situational awareness, helping ensure that local offloading choices remain consistent with system-wide learning objectives. Overall, the proposed LLM-assisted decision intelligence layer integrates predictive modeling, distributed learning, and control of execution into a single inference-centric pipeline. By fusing DT-driven forecasting, federated coordination cues, and distilled MARL expertise, the framework enables energy-aware, latency-sensitive offloading while significantly reducing operational complexity compared to traditional RL-based approaches.

To make the LLM-based decision-making process explicit and reproducible, we formally specify the structure of the contextual prompt. Table 5.2 outlines the individual elements that make up the prompt and explains the functional role of each source of information in the transfer decision. By distinguishing between instantaneous device state, DT-driven short-term predictions, FL indicators, and offline MARL examples, the prompt design provides the LLM with a compact yet comprehensive view of both local constraints and the global system context.

Figure 5.3 illustrates the corresponding prompt composition workflow. Multiple heterogeneous signals originating from the DT, sensor node state, federated aggregation process, and the offline reinforcement learning knowledge base—are first

Table 5.2: Structure of the LLM prompt and role of each contextual component.

Prompt Component	**Source**	**Role in Decision-Making**
Current device state	Sensor node	Encodes instantaneous constraints such as battery level, CPU load, and queue occupancy.
DT-predicted resource trends	Digital Twin	Provides short-term forecasts of battery drain, CPU saturation, and latency, enabling proactive rather than reactive offloading.
Network condition summary	DT / monitoring	Captures expected communication cost and congestion likelihood, influencing offload feasibility.
FL indicators	FL aggregator	Reflects global model convergence and training stability, discouraging actions that disrupt collaborative learning.
Offline MARL exemplars	RL knowledge base	Supplies reference offloading cases with high reward under similar conditions, guiding inference via few-shot context.

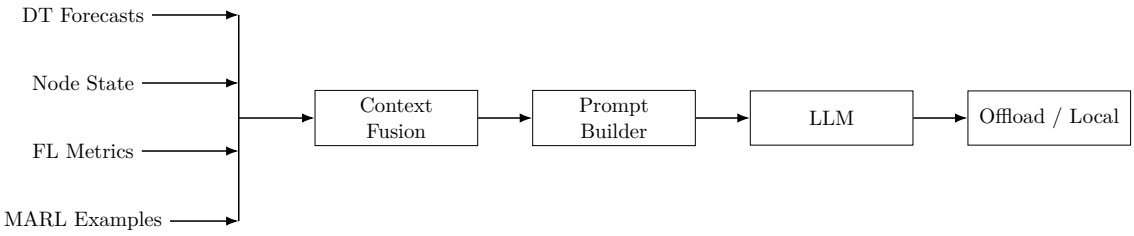


Figure 5.3: LLM-assisted offloading decision flow.

consolidated by a dedicated prompt builder before being passed to the LLM controller. This design enforces a clear separation between data acquisition, context synthesis, and decision inference, thereby avoiding direct coupling between learning components and execution logic. As a result, offloading decisions emerge from structured reasoning over predictive and historical context rather than from ad hoc rule-based heuristics or online policy training.

The next chapter presents the experimental setup and evaluation framework used to validate the proposed approach. It describes the simulation environment, system configurations, evaluation metrics, and test scenarios designed to systematically assess performance, robustness under diverse operating conditions.

Chapter 6

Simulation Setup and Evaluation

This chapter describes the experimental setup designed to evaluate the proposed XDT-FMARL framework. The evaluation pursues three main goals: (i) verifying the system's ability to operate efficiently under heterogeneous IoT conditions, (ii) measuring performance gains relative to standard offloading baselines, and (iii) examining the robustness, scalability, and interpretability of the proposed decision-making pipeline.

Given the inherently dynamic and distributed nature of IoT ecosystems, the experimental approach is structured to emulate realistic device diversity, random network conditions, time-critical workloads, and operation under limited resources. A discrete-event simulation framework is used to preserve detailed control over system parameters while still capturing the asynchronous behavior characteristic of real-world deployments.

The evaluation framework combines three closely linked elements: FL for joint model training without exposing raw data, predictive DTs for near-term resource prediction, and LLM-driven reasoning for dynamic task offloading. Instead of examining these components separately, this chapter evaluates their combined influence on energy consumption, latency, decision accuracy, and overall system robustness. The remainder of this chapter is structured as follows:

- Section 6.1 outlines the setup of the simulation environment, detailing the IoT device model, the workload generation method, the network latency behavior, DT synchronization, FL coordination and the LLM-assisted offloading

pipeline.

- Section 6.2 specifies the evaluation metrics used to quantify system performance, covering energy retention, task latency, offloading patterns, learning stability, and explainability measures.
- Section 6.3 outlines the operational scenarios used to assess the framework under different conditions, such as changes in network stability, workload intensity, device heterogeneity, and LLM decision contexts.

Table 6.1 summarizes the roles and implementation details of each module within the proposed pipeline, clarifying how data flows between system components during runtime decision execution.

Through systematic experiments in controlled settings, this chapter provides empirical support that the proposed framework can deliver proactive, energy-aware, and globally consistent task scheduling in heterogeneous IoT environments.

6.1 Simulation Design

The simulation environment is implemented in `Python` using a Jupyter Notebook framework, with discrete-event modeling conducted through the `SimPy` library. SimPy enables precise representation of asynchronous processes and distributed time evolution, which are essential characteristics of real-world IoT deployments. Each IoT device and edge server is instantiated as an independent SimPy process operating under its own logical timeline. This design allows the simulation to capture decentralized execution, heterogeneous processing delays, fluctuating workloads, and independent control decisions without imposing artificial synchronization constraints.

The simulated topology consists of $N = 12$ heterogeneous IoT devices and $K = 2$ edge servers. The IoT devices are characterized by limited computational capacity, finite battery reserves, and time-varying wireless connectivity. Edge servers serve a dual role as computational offloading endpoints and federated aggregation nodes. This topology reflects practical edge-assisted IoT infrastructures in which devices must dynamically decide whether to execute tasks locally or delegate computation to nearby edge resources.

Table 6.1: Pipeline components and their roles within the XDT–FMARL workflow.

Component	Inputs	Outputs	Role in Framework	Implementation Details
IoT Devices	state, task descriptors	execution logs, local updates	Execute tasks locally or offload them to edge resources while continuously reporting runtime device status.	SimPy-based node processes with analytical energy consumption modeling.
Digital Twin	historical device telemetry	short-term resource forecasts	Predict near-future variations in battery level, CPU utilization, and communication latency to enable proactive scheduling decisions.	Regression-based prediction combined with moving-average smoothing.
Federated Learning	local model updates	global aggregated model	Coordinate privacy-preserving distributed training across heterogeneous IoT nodes without transmitting raw sensor data.	FedAvg aggregation with convergence monitoring statistics.
Prompt Builder	DT forecasts, FL convergence metrics, node state	structured prompt representation	Fuse predictive and collaborative learning signals into structured contextual inputs for the LLM controller.	Template-driven prompt construction with exemplar-based formatting.
LLM Controller	structured prompt	offloading decision	Determine whether tasks should be executed locally or offloaded based on predicted resource availability and system objectives.	Inference-only execution with temperature = 0 for deterministic responses.

6.1.1 Task and Workload Modeling

Each IoT device periodically generates computational tasks designed to emulate realistic IoT workloads. Task input sizes are sampled within the range of 64 KB to 512 KB, while computational requirements vary between 0.5 and 3.0 GFLOPs. Task deadlines are drawn from the interval $[150, 400]$ ms, modeling latency-sensitive applications such as industrial monitoring, vehicular telemetry, and healthcare sensing.

This workload configuration ensures diversity in computational intensity and temporal urgency, thereby enabling evaluation of offloading strategies under heterogeneous real-time constraints. The stochastic task generation process ensures that the system operates under continuously varying demand, reflecting realistic operational volatility.

6.1.2 Data Handling and Battery Modeling

All sensor readings, task-related information, and time-stamped records are initially aggregated into a structured `Pandas` DataFrame, which serves as a unified container for enforcing chronological consistency across the entire simulation horizon. All timestamp fields are converted into standardized `datetime` objects, allowing accurate temporal alignment among sensing events, task creation, execution windows, and energy state updates. Maintaining this level of temporal coherence is crucial not only for faithfully representing the evolving battery dynamics, but also for preserving tight synchronization between the simulated physical processes and their corresponding digital representations throughout the experiment.

Energy usage is estimated with a lightweight, regression-based method implemented using the `scikit-learn` library. To reflect realistic deployment scenarios, we create synthetic training data by sampling CPU utilization levels and execution times over controlled intervals that resemble common IoT workload patterns. This dataset is then used to fit a linear regression model that learns how computational load and processing duration jointly influence the resulting battery consumption, enabling us to approximate device energy drain under varying operating conditions.

During simulation, the trained regression model continuously estimates energy usage at every time step based on the device's instantaneous CPU load and the du-

ration of the tasks being executed. The resulting energy drain estimate is deducted from the current battery charge, enabling ongoing monitoring of how the battery depletes over time. In parallel, communication-induced energy expenditures are incorporated into this update, so that both on-device computation and remote offloading overheads are captured in a unified manner. This combined treatment ensures that battery dynamics unfold in a realistic way throughout the simulation horizon, supporting a more reliable assessment and comparison of energy-aware offloading strategies when the system is subjected to prolonged and intensive workloads.

6.1.3 Network Dynamics and Communication Modeling

Wireless communication latency is modeled as a stochastic process to capture the time-varying and uncertain behavior of practical IoT networks. At every simulation step, the nominal latency value is drawn from a Gaussian distribution with a mean of 30 ms and a standard deviation of 3 ms. This setup is intended to emulate moderate network conditions that are common in edge-assisted IoT scenarios, in which latency remains centered around a relatively stable average while naturally fluctuating due to varying channel quality, interference, and background traffic load.

To better reflect short-lived congestion conditions, we incorporate an additional delay term in a probabilistic manner. Specifically, with a small probability of 5%, we add an extra latency component, drawn from a uniform distribution over $[5, 10]$ ms, on top of the baseline delay. This stochastic extension is intended to emulate occasional congestion spikes, interference effects, or brief routing anomalies that are commonly observed in real-world wireless networks and can momentarily degrade performance.

At each simulation step, the overall latency is calculated as the sum of two terms: a Gaussian-distributed baseline delay and an additional congestion-dependent delay. This mixed stochastic design enables the simulator to capture both typical, steady-state communication patterns and sporadic slowdowns. By introducing controlled randomness together with infrequent but significant congestion events, the model produces delay trajectories that are realistic and diverse enough to rigorously exercise adaptive offloading strategies, yet remain statistically consistent and

reproducible across multiple experimental runs.

6.1.4 Digital Twin Synchronization and Forecasting

Within the proposed framework, every IoT node hosts a lightweight DT instance that serves as a live virtual representation of the corresponding physical device. This DT is realized as a continuously executing simulation module that, at each simulation step, exchanges state information with its physical counterpart to remain closely synchronized. As part of this synchronization cycle, the DT collects and updates key operational metrics, such as the current remaining battery charge, the instantaneous processor (CPU) load, the wireless communication latency experienced, and the length of the local queue of pending tasks waiting to be processed.

DT integrates continuous monitoring with short-term forecasting. Battery behavior is estimated by the previously trained linear regression model, which projects near-future energy consumption from the current CPU load together with the anticipated execution duration. Concurrently, CPU utilization is smoothed using a moving-window average, implemented via a deque-based structure, producing a more stable estimate of the imminent computational workload. In parallel, network latency values are sampled from the previously defined stochastic model, and their short-range dynamics are also incorporated into the predictive state of DT. In this way, the DT jointly captures and updates both computational demand and communication conditions, so that its internal representation remains aligned with the evolving system behavior.

By integrating regression-based battery prediction with workload leveling and latency prediction, the DT generates short-horizon estimates of both resource availability and communication quality. These estimates capture projected battery discharge patterns, possible CPU overload situations, and likely fluctuations in latency for the upcoming decision interval. The resulting forecasted indicators are subsequently communicated back to the associated IoT node, thereby maintaining a two-way exchange of information between the physical device and its virtual counterpart.

Beyond simple monitoring, the DT functions as an abstraction layer between

low-level telemetry data and the higher-level offloading controller. Instead of forwarding only raw instantaneous measurements, it generates structured predictive summaries that are directly integrated into the LLM-based decision prompts. This architecture allows the controller to reason about both the current operational state and the likely evolution of system dynamics over a future horizon. Consequently, offloading decisions can be made proactively rather than reactively to recent changes, which in turn enhances energy efficiency, mitigates latency-related risks, and helps to maintain stable learning performance even when workloads and network conditions are highly variable or congested.

6.1.5 Federated Learning Coordination

To support collaborative intelligence without compromising data locality, the framework uses an FL-based synchronization scheme that runs in parallel with DT-driven forecasting and LLM-powered decision control. Each IoT node hosts a compact local ML model that is trained solely on its own physical and virtual sensor data streams. By adopting this decentralized learning approach, raw sensor readings never leave the device, which safeguards privacy, reduces bandwidth consumption, and lowers communication overhead during model coordination.

In each federated round, devices conduct local training for a fixed number of epochs on their own private data. The local models are typically linear predictors or shallow neural networks, chosen to remain computationally practical for resource-limited IoT devices. After local training finishes, only the model parameters (or gradients) are sent to an edge-level federated aggregator. No raw sensor data or task-specific information is shared.

The aggregator executes synchronous FedAvg, merging the uploaded local model parameters into a single global model, where the contribution of each device is weighted by the proportion of data it provides. This updated global model is then distributed to all participating nodes and used to initialize their next round of local training. By repeatedly performing these synchronization rounds, the global model gradually becomes more accurate and resilient, even in the presence of heterogeneous devices and non-IID data distributions across clients.

Beyond improving predictive accuracy, the FL process exposes system-wide convergence patterns that enhance adaptive offloading decisions. Metrics such as the progression of the loss, the stability of convergence, and the consistency of model updates are derived from the federated training rounds and fed into the contextual information available to the decision controller. As a result, devices no longer base their actions only on momentary local resource conditions, but they also coordinate their choices with the broader learning behavior and trends observed across the entire network.

This synchronization mechanism builds a coordinated learning layer that enhances and supports the DT's short-term predictions. The DT offers fine-grained, localized forecasts of how resources will evolve, while FL delivers system-wide knowledge aggregation and convergence feedback across all devices. By operating together, these layers guide offloading decisions so they account for both local resource limitations and global learning goals, thereby sustaining scalability, strengthening robustness, and preserving privacy throughout the distributed IoT environment.

6.1.6 LLM-Assisted Offloading Execution

The proposed framework adopts an LLM as the central runtime decision engine for adaptive task offloading within the federated IoT environment. The LLM operates through structured in-context reasoning. This eliminates the need for explicit policy optimization while preserving adaptability across heterogeneous and dynamically changing network states. The LLM inference layer runs within the Ollama runtime in CPU mode, using a locally hosted `llama3` model instance. This setup supports controlled experimentation without external API dependencies and provides consistent, reproducible latency measurements.

At each decision point, the LLM is provided with a structured description of the operating conditions of the device, covering the current status of the resource, the short-term predictions of DT, the measured or estimated communication delays, and the selected metrics of the FL synchronization phase. Instead of using a fixed, pre-trained control policy, the LLM applies prompt-based inference to these contextual data to decide whether the arriving task should be processed on the device or

Table 6.2: LLM prompt signals, decision roles, and inference latency.

Prompt Signal / Field	**Type / Units**	**Source**	**Purpose in Offloading Decision**
Predicted battery level (b_{pred})	% (0–100)	DT forecast	Guides offloading when energy reserves are low; prevents critical battery drain under sustained workloads.
Predicted CPU utilization (c_{pred})	fraction (0–1)	DT forecast	Helps LLM anticipate processing saturation; avoids assigning local tasks when compute load exceeds thresholds.
Predicted network latency (ℓ_{pred})	ms	DT forecast	Enables proactive redirection of tasks when wireless congestion is likely.
Predicted queue occupancy (q_{pred})	tasks / slots	DT forecast	Indicates backlog; promotes offloading when local queues risk violating deadlines.
FL global loss delta (ΔL_{FL})	relative error	Federated aggregator	Signals whether the global model is stabilizing or diverging; helps maintain consistent learning performance.
FL accuracy delta (ΔA_{FL})	% improvement	Federated aggregator	Encourages balanced offloading when convergence slows; prioritizes devices contributing novel data.
Exemplar scenarios ($\mathcal{E}$)	structured list	MARL offline cache	Provides high-reward reference policies for states similar to the current one; reduces uncertainty in novel conditions.
LLM Inference Latency (Ollama `llama3`, CPU mode, N=12, K=2):			
Mean decision latency	∼178 ms per prompt, measured over 60 epochs.		
95th percentile latency	∼262 ms, corresponding to larger prompt contexts (more exemplars).		
Prompt token count (avg.)	∼680 tokens (≈1.2 kB), constant-time cached MARL lookups.		

offloaded to a suitable edge server. Table 6.2 details the structured prompt signals, their origin within the XDT-FMARL pipeline, and their functional role in the offload decision process, together with the measured inference characteristics under the deployed configuration.

This reasoning strategy makes it possible to fuse diverse types of input into a single, coherent decision-making pipeline. The LLM silently weighs the trade-offs between energy conservation, computational workload distribution, and communication latency, and does so without relying on gradient-based online learning or constant parameter updates. As the number of devices, wireless channel conditions, and task loads change over time, the decision process remains robust and consistent because it is grounded in real-time contextual information rather than in static policies tuned to a specific, fixed operating environment.

By integrating DT-based predictive awareness with FL-generated global learning

signals, the offloading procedure evolves from a solely reactive scheduling scheme into a forward-looking, context-sensitive control paradigm. In this architecture, the LLM functions as an adaptive coordination layer, fusing localized resource predictions with system-wide intelligence to drive real-time execution choices. The following subsection provides an expanded discussion of the technical underpinnings of this approach, including prompt engineering strategies, methods for structured context representation, techniques for embedding convergence-related feedback, and the use of offline MARL-based exemplars to refine decision quality and robustness.

6.1.7 Node Execution Flow

In the proposed SimPy-based simulation environment, every IoT device is modeled as an independent process, capturing the asynchronous, distributed, and event-driven nature of real IoT deployments. This process-based design enables nodes to function autonomously while interacting with shared components such as the DT, the FL aggregator, and the LLM-driven offloading controller.

At each simulation step, every node refreshes its operational state using three main procedures: CPU load sampling and smoothing, battery drain estimation, and stochastic network latency generation. CPU usage is measured and averaged over a short moving window to capture brief workload variations while retaining short-term trends. Battery consumption is then inferred with the lightweight regression-based energy model, which incorporates both computational intensity and execution time. Simultaneously, wireless latency is produced using the previously defined stochastic network model, allowing the simulation to reflect typical communication conditions as well as sporadic congestion events.

These updates together specify the node's state representation, comprising its residual battery charge, current CPU usage, estimated network latency, and short-term average load. This state is stored locally and provides the foundation for both prediction and decision-making.

After revising its internal state, the node aligns with its linked DT module. The DT generates short-term predictions of resource dynamics, such as expected battery consumption and upcoming CPU usage patterns. These forecasts expand

the node's view beyond real-time readings and support anticipatory control. At the same time, convergence metrics and synchronization data from the FL process add global context that captures the overall learning progress across the network.

All relevant contextual information is compiled into a structured runtime prompt and passed to the LLM-based controller. Using this comprehensive state description, the LLM analyzes current conditions such as workload, latency requirements, and resource availability and then issues an offloading directive, selecting local processing on the node or remote execution at the edge. Upon receiving this directive, the node immediately enforces it by reorganizing its task queue, reallocating compute and memory resources, and updating its energy usage tracking to reflect the new execution strategy.

Through this closed feedback loop linking local state updates, DT-based prediction, FL-level coordination, and LLM-supported reasoning, each sensor node functions as a fully self-contained adaptive control entity. In this architecture, task scheduling decisions are governed not only by instantaneous system metrics, such as current workload, latency, and energy availability, but also by predicted resource trajectories and overarching global learning goals. As a result, the simulation faithfully represents the coupled evolution of computation, communication, and distributed intelligence across diverse IoT devices and network conditions, providing a robust and systematic basis for analyzing, comparing, and rigorously evaluating adaptive task offloading policies.

6.2 Evaluation Metrics

To evaluate the performance of the proposed framework, we introduce a set of quantitative metrics that reflect system efficiency, responsiveness, learning stability, and consistency of decisions. All reported metrics are computed per episode and averaged over multiple independent runs. Table 6.3 summarizes the quantitative metrics employed to evaluate system efficiency, responsiveness, learning stability, and decision alignment across all experimental scenarios.

6.2.1 Latency and Responsiveness

Average Latency (ms) denotes the mean time to complete tasks across all devices in an episode. This metric captures how responsive the offloading strategy is under different workload and network conditions.

Deadline Violation Rate is the fraction of tasks that fail to meet their latency constraint relative to all processed tasks. It measures how effectively the controller fulfills real-time requirements.

6.2.2 Energy Efficiency and Stability

Energy Consumption (J) denotes the total battery usage over an episode, accounting for both on-device computation and communication overhead.

Mean Residual Battery Level (%) denotes the average remaining battery charge of all devices at the conclusion of each episode.

Battery Stability Index is defined as the standard deviation of the batteries' final charge levels over all episodes. This measure captures how consistently energy is preserved over time and assesses whether the policy sustains stable long-term behavior instead of exhibiting fluctuating consumption.

6.2.3 Offloading Behavior and Adaptivity

Offload Ratio represents the share of tasks processed at the edge compared to the overall number of tasks.

Offload Responsiveness quantifies how the offload ratio shifts between light and heavy workload conditions. This metric assesses whether the controller dynamically adjusts its scheduling strategy in reaction to changing environmental conditions.

6.2.4 Learning and Convergence Behavior

Convergence Episode Index denotes the episode beyond which performance metrics (battery level and offload count) remain stable within a specified tolerance range. This measure characterizes how efficiently and how quickly the system converges.

Decision-Match Accuracy (%) evaluates how often LLM-produced actions coincide with those of the trained D3QN expert policy under the same system states, quantifying the consistency between in-context reasoning and reinforcement learning–based expertise.

Federated Convergence Stability tracks relative variations in global loss and accuracy over federated rounds, evaluating the consistency of collaborative learning under non-IID data distributions.

6.2.5 Interpretability Assessment

Saliency Dominance Score measures the share of decisions where key state variables (battery level, CPU load, or latency) are included among the highest gradient magnitudes. This metric checks whether the model's decision rationale matches expected system priorities, thereby supporting interpretability.

6.3 Test Scenarios

6.3.1 Baseline Strategy Comparison

To establish a controlled reference for evaluating adaptive offloading methods, we implement two deterministic scheduling schemes as baseline configurations: (i) Naive Offload and (ii) Pure Local Processing. These baselines reflect opposing extremes of task assignment behavior and help isolate the effect of adaptive decision-making in the proposed framework.

Naive Offload Strategy

In the naive offload setup, each device forwards all incoming tasks to the edge server, ignoring factors such as local battery status, CPU load, network delay, or current workload. This approach presumes that remote execution is always better and eliminates any context-aware decision-making.

As a result, tasks are offloaded at every decision point, leading to a rigid and fully deterministic offloading pattern. Although this removes any local computational load, it causes ongoing communication and overlooks possible transmission

Table 6.3: Summary of evaluation metrics used for performance assessment.

Metric	Unit	Description	Objective
Average Latency	ms	Mean task completion delay per episode.	Minimize
Deadline Violation Rate	Ratio (0–1)	Fraction of tasks exceeding predefined latency constraints.	Minimize
Energy Consumption	Joules	Total computation and communication energy used during execution.	Minimize
Mean Residual Battery	%	Average final battery level observed across devices per episode.	Maximize
Battery Stability Index	Std. Dev. (%)	Variance of final battery levels across episodes.	Stabilize
Offload Ratio	Ratio (0–1)	Proportion of tasks executed at edge nodes rather than locally.	Balance
Offload Responsiveness	Relative Change	Variation of offload ratio across workload regimes.	Adapt
Convergence Episode Index	Episode #	Episode after which policy performance stabilizes.	Minimize
Decision-Match Accuracy	%	Agreement between LLM and expert D3QN decisions.	Maximize
FL Convergence Stability	Relative Delta	Variation of global loss/accuracy across FL rounds.	Stabilize
Saliency Dominance Score	%	Frequency of critical features dominating gradient saliency.	Validate

overhead. Thus, the naive scheme represents a purely transmission-oriented scheduling approach without adaptive resource allocation.

Pure Local Processing Strategy

In contrast, the pure local baseline completely disables offloading. Every computational task is handled directly on the device, regardless of workload intensity or network state. As a result, there is no communication energy cost for task transfer.

This strategy represents a fully self-contained processing model, where decisions are made without relying on external computational resources. While it eliminates transmission overhead, it fails to consider cases where distributing the workload might be advantageous.

Role of Baselines in Evaluation

Together, these two baselines define upper and lower behavioral limits on offloading frequency. The naive offload strategy reflects always delegating, while the pure local strategy reflects always keeping tasks in-house. Neither uses predictive insight, collaborative learning cues, or adaptive reasoning.

By comparing the proposed framework's decision mechanism with these extreme cases, the evaluation framework can determine whether adaptive offloading attains a balanced compromise between communication overhead and local computation, instead of relying on fixed scheduling rules.

6.3.2 Single-Agent RL Baseline

Alongside the static baselines, we implement a centralized single-agent RL setup as a learning-based reference model. This baseline enables assessment of how decentralized federated coordination and LLM-assisted reasoning perform relative to a standard centralized RL controller operating in the same simulated IoT environment.

Architecture and Training Setup

The single-agent baseline employs a D3QN architecture with PER and multi-step returns. Unlike the federated multi-agent configuration, a single centralized agent interacts with the entire network of simulated devices and learns a global policy across all nodes.

Each episode consists of 35 discrete time steps, and training is conducted over 350 episodes to maintain consistency with the main experimental configuration. The state representation, battery depletion model, and latency simulation remain identical to those used in the proposed framework to ensure comparability between experiments.

To evaluate robustness to stochastic initialization, the model is trained under multiple random seeds. This setup enables examination of the behavior of the policy convergence under different random starting conditions without altering the environmental parameters.

Reward and Exploration Configuration

The reward formulation follows the same structural objectives as the federated RL configuration, balancing energy preservation, latency awareness, and offloading penalties. However, exploration and optimization are managed centrally rather than independently per node. An exploration strategy ϵ is used with controlled decay throughout the training episodes, where ϵ represents the probability that the agent chooses a random action rather than the optimal one inferred from its current knowledge. Target network updates and replay buffer sampling are performed according to the standard D3QN practice, and all hyperparameters are selected to ensure stable convergence within the constrained simulation horizon. Target network updates and replay buffer sampling are performed according to the standard D3QN practice. All hyperparameters are selected to ensure stable convergence within the constrained simulation horizon.

Role in Comparative Evaluation

The purpose of this baseline is to isolate the effect of decentralization and collaborative learning. By training a single agent over the entire system state, this configuration removes inter-agent heterogeneity and federated aggregation effects. It therefore serves as a controlled comparison point for assessing:

- The impact of decentralized multi-agent coordination,
- The contribution of FL synchronization,
- The influence of structured prompt-based reasoning in the LLM controller.

This baseline does not incorporate DT-driven predictive augmentation or LLM-assisted decision inference. Instead, it relies solely on policy optimization through iterative environment interaction, reflecting a conventional RL-based offloading controller.

6.3.3 LLM Controller Behavior Under Diverse IoT Conditions

To consistently evaluate controller performance across different operating conditions, four controlled experimental scenarios were established. In each scenario, selected environmental parameters are altered, while the simulation setup, episode duration, and state representation remain unchanged. This configuration supports fair cross-controller comparisons under increasingly demanding conditions.

Scenario 1: Nominal Operating Conditions

The baseline scenario reflects standard IoT operation. Task arrival rates are moderate and stable over time, wireless latency follows its usual stochastic distribution without increased variance, and all IoT devices possess similar computational resources. This setup acts as a reference for evaluating controller performance under steady-state conditions.

Scenario 2: High Workload Intensity

To simulate bursty or computation-heavy conditions, task arrival rates are raised compared to the baseline setup. This leads to fuller queues, higher CPU usage, and greater scheduling pressure. All other environmental parameters are kept constant, isolating the impact of workload intensity on decision-making behavior.

Scenario 3: Heterogeneous Device Capabilities

In this setting, IoT nodes are given diverse computational capabilities and resource configurations. Variations in CPU frequency, processing performance, and battery discharge behavior are incorporated to reflect realistic deployment heterogeneity. Network conditions and workload patterns are kept the same as in the baseline, enabling assessment of how controllers cope with differences across devices.

Scenario 4: Unstable Wireless Network Conditions

To evaluate robustness under communication uncertainty, wireless latency behavior is adjusted to feature greater variance and more frequent delay spikes. Bandwidth variability and sporadic congestion events are modeled to mirror real-world wireless instability. Task generation rates and device capabilities are kept constant to isolate the impact of network conditions.

The following chapter reports the experimental results and examines how the proposed framework performs under various operating conditions. The analysis considers energy efficiency, task latency, offloading patterns, and decision interpretability to evaluate the effectiveness and robustness of the proposed approach in realistic IoT scenarios.

Chapter 7

Results and Discussion

This chapter presents and examines the experimental results of the proposed framework for proactive task offloading in distributed IoT settings. The evaluation centers on how effectively the system manages the trade-off between two key goals: reducing computational latency and conserving device energy. In highly resource-limited IoT networks, these goals naturally conflict. Processing tasks locally saves communication energy but can increase computation time, whereas extensive offloading to edge servers lowers latency at the expense of higher communication energy consumption.

The results presented in this chapter, therefore, assess how well the proposed framework manages this trade-off across a range of operating conditions. Specifically, the evaluation examines whether combining DT predictive modeling, FL coordination, and LLM reasoning supports a proactive offloading strategy that enhances both responsiveness and energy efficiency relative to conventional methods.

The analysis examines several experimental conditions, such as stable networks, high load intensity, heterogeneous device resources, and unreliable wireless links. Under these scenarios, the proposed LLM-assisted controller is evaluated against multiple baselines, including purely local execution, greedy threshold-based offloading, and fully edge-based processing. These comparisons make it possible to isolate the performance improvements that stem specifically from the LLM controller's contextual reasoning capabilities.

7.1 Proactive Task Offloading Performance

This section assesses how effectively the proposed framework delivers proactive, context-aware task scheduling. The evaluation centers on two key performance metrics: energy efficiency and latency reduction. Combined, these metrics reflect the system's capability to sustain device operation while ensuring prompt task execution.

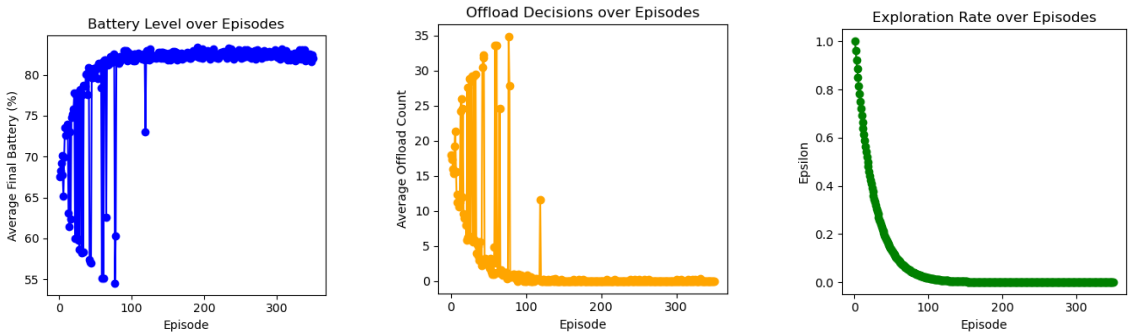
7.2 RL Policy Convergence and Training Behavior

To evaluate the learning dynamics of the proposed reinforcement learning controller, two primary indicators are analyzed throughout training: the final battery level achieved at the end of each episode and the frequency of offloading actions performed by the agent. The primary design objective of the reward function is to maintain the battery level of the device above 85% while still allowing occasional task loading when local computational pressure increases.

Figure 7.1 illustrates the evolution of three training indicators across 350 episodes. Specifically, Figure 7.1(a) reports the average final battery level between devices, Figure 7.1(b) presents the average number of offloading actions per episode and Figure 7.1(c) shows the exploration rate (ϵ). Collectively, these indicators provide insight into how the RL agent transitions from exploratory behavior to a stable offloading strategy.

During the early training stages, the agent explores the action space extensively due to the high exploration rate. This results in relatively large variations in both offload frequency and battery levels. In some episodes the battery level temporarily falls below the desired 85% threshold, which occurs when exploratory offloading actions introduce additional communication costs under unfavorable conditions.

As training progresses, the exploration rate gradually decreases, allowing the agent to exploit the most beneficial decision patterns it has discovered. Around episode 100, the curves begin to stabilize, indicating that the learning process has converged toward a consistent decision policy. From this point onward, the controller



(a) Battery Level (b) Offload Decision (c) Exploration Rate (ϵ)

Figure 7.1: RL training dynamics showing battery retention, offload frequency, and exploration decay.

maintains battery levels above the target threshold while using offloading selectively to mitigate local CPU load.

The observed convergence behavior confirms the effectiveness of the D3QN architecture combined with PER and multi-step return updates. These mechanisms accelerate learning by emphasizing informative experiences and propagating reward signals more effectively across decision sequences.

Overall, the learned policy demonstrates a balanced operational strategy. Instead of aggressively offloading or strictly processing tasks locally, the controller adopts a cautious offloading approach that preserves energy while still benefiting from distributed computational resources when necessary. The stability of the curves after convergence further indicates that the resulting policy is robust to variations in CPU demand, wireless latency, and initial battery conditions encountered during training.

7.3 Baseline Strategy Comparison

To interpret the behavior of the learned RL policy, two reference scheduling strategies are used as baseline comparisons. These baselines reflect extreme operational policies that deliberately ignore contextual information about the system state. This comparison thus emphasizes the benefits of adaptive decision-making in distributed IoT settings.

The first baseline is a *Naive Offload* strategy, where every incoming task is immediately sent to the edge server. The second baseline is a *Pure Local Processing*

strategy, in which all tasks are handled locally and offloading is entirely disabled. Together, these two strategies define opposite ends of the scheduling spectrum and serve as useful reference points for assessing the performance of the proposed adaptive controller.

7.3.1 Naive Offload Strategy

Under the naive offloading strategy, each IoT device forwards all arriving tasks to the edge server, irrespective of its current battery state, CPU load, or network latency. Although this fully relieves the device of local computation, it incurs a steady communication overhead. As shown in Figure 7.2 (a), the average final battery level stabilizes at roughly 69%–72%, reflecting the impact of aggressive offloading on total energy consumption due to transmission costs. This is consistent with the offloading behavior observed in as shown in Figure 7.2 (b), where the offloading rate remains nearly unchanged throughout training, holding constant at 35 offloads per episode. While local CPU usage is reduced, this persistent offloading pattern is the primary driver of the elevated energy consumption.

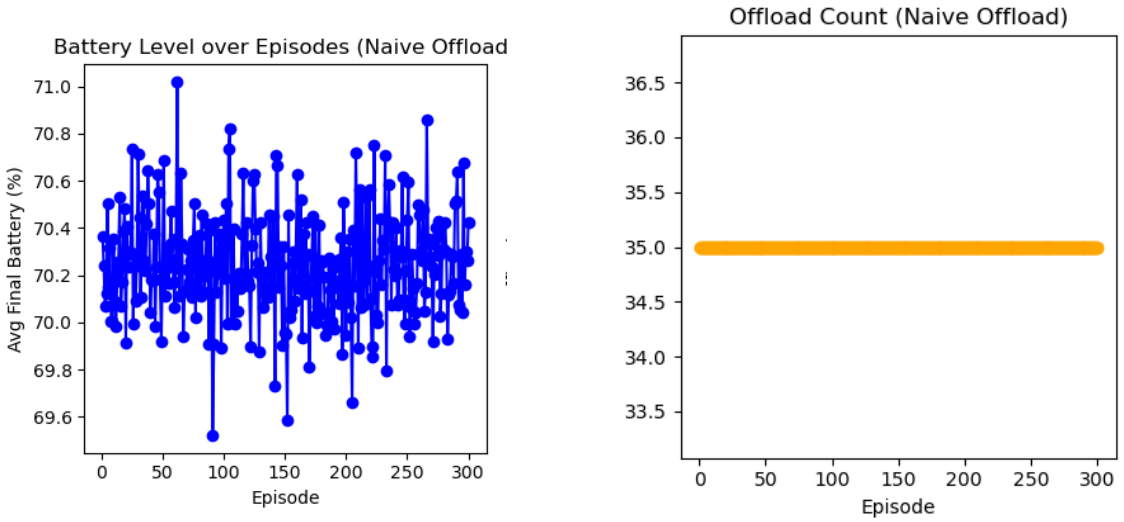
These findings indicate that blindly offloading tasks does not inherently improve energy efficiency, as the extra communication overhead can negate the computational energy savings. This baseline thus underscores the necessity of incorporating contextual resource information into offloading decisions.

7.3.2 Pure Local Processing Strategy

The second baseline reflects the opposite operational extreme: every task is processed locally on the device, and offloading is never used. Consequently, the offload count remains zero for all episodes as shown in Figure 7.3 (a).

Since no wireless transmission takes place, energy use stems exclusively from the device's CPU. As illustrated in Figure 7.3 (b), the average final battery level settles around 90–91%, showing that eliminating communication overhead substantially lowers energy consumption.

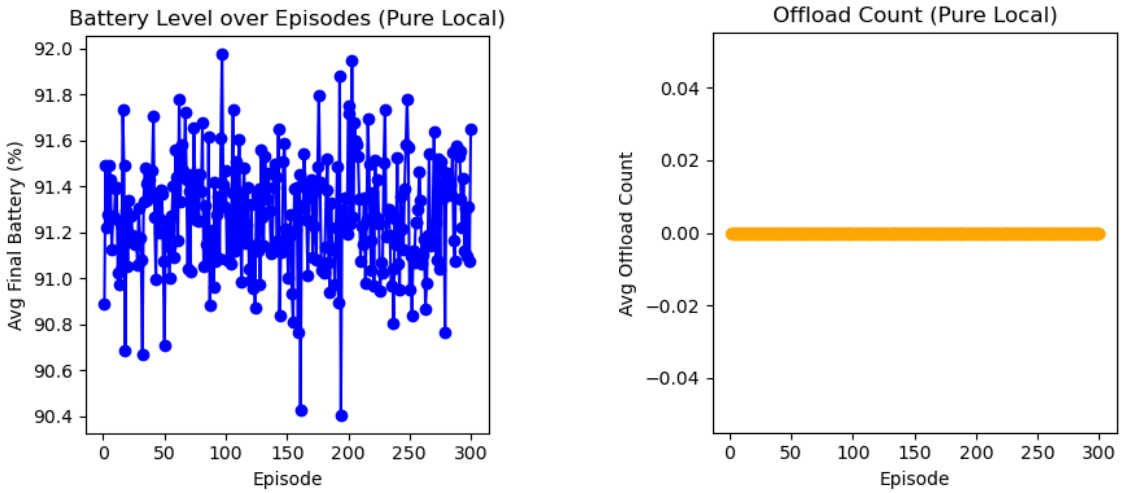
Yet, this approach does not leverage distributed computing. When local CPU load is high or network conditions would support efficient offloading, the system



(a) Battery Level (b) Offload Decision

Figure 7.2: Performance characteristics of the naive offload strategy.

cannot shift tasks elsewhere. Thus, the pure local strategy gives up computational flexibility and can result in higher processing latency under heavy workloads.



(a) Battery Level (b) Offload Decisions

Figure 7.3: Performance characteristics of the pure local processing strategy.

7.3.3 Comparative Insights

The comparison of the two baseline strategies with the learned RL policy highlights several key insights.

First, the naive offloading strategy shows that delegating too many tasks in-

creases communication overhead and, in turn, shortens battery life. While it does reduce on-device computation, the constant data transfer cost ultimately dominates any gains.

Second, the pure local strategy maximizes battery life by eliminating transmission costs. However, this choice prevents the system from exploiting distributed computing capabilities and limits its adaptability under changing workloads.

Third, the RL-based controller settles on an operating point between these two extremes. Rather than always offloading or never offloading, the learned policy offloads selectively, only when the anticipated benefit exceeds the communication overhead. This balanced behavior conserves battery power while still utilizing edge resources when it is beneficial.

Taken together, these baselines indicate that the best task scheduling strategy in IoT systems lies between full local processing and full offloading. Context-aware decision policies that account for battery level, CPU utilization, and network conditions are crucial for enabling energy-efficient and scalable distributed intelligence.

7.4 Performance Under Reference Operating Conditions

Table 7.1 summarizes the system performance in stable network conditions with uniform IoT devices. In this setup, the LLM-based controller attains latency figures comparable to full edge offloading while consuming substantially less energy. This suggests that contextual reasoning driven by structured prompts can effectively replace computationally intensive online optimization methods.

The LLM controller bases its decisions on several contextual cues encoded in the prompt, such as DT resource predictions and FL convergence metrics. By jointly analyzing these inputs, it offloads only those tasks that are likely to yield significant latency gains without leading to high communication energy consumption. This predictive approach cuts down on redundant task transfers and avoids the oscillatory patterns commonly seen in naive greedy or simple threshold-based methods.

The findings further highlight the limitations of the two extreme baseline strate-

gies. Processing entirely on the device remains the most energy-saving option, as it avoids any wireless data transmission; yet, this configuration is unable to meet strict latency requirements once the computational demand becomes substantial. At the other extreme, offloading all tasks to the edge reliably yields the shortest response times, but this benefit comes with a significant increase in communication-related energy consumption. Taken together, these observations show that the proposed framework, through prompt-driven reasoning, identifies an intermediate operating regime that more effectively balances responsiveness with the preservation of device energy resources, thereby avoiding the inefficiencies inherent in both extremes.

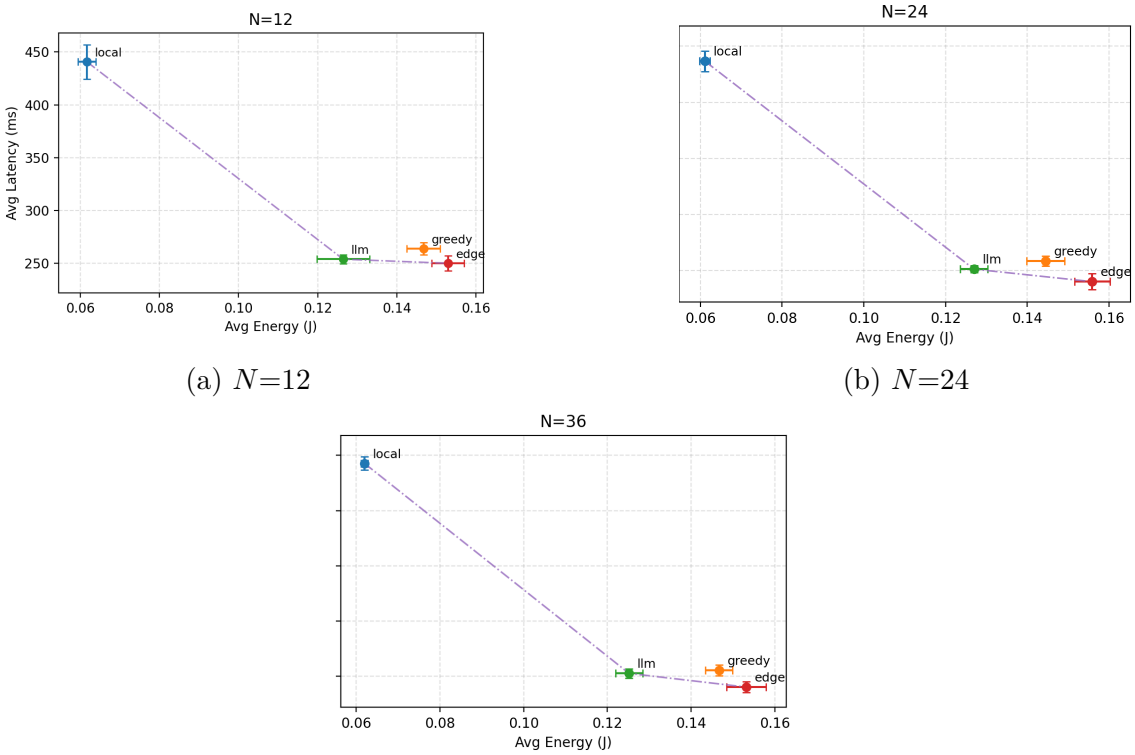
Figure 7.4 provides a more detailed view of how energy consumption and latency trade off across three distinct network sizes. The *local* baseline lies in the region with relatively high latency but very low energy expenditure, underscoring the computational constraints inherent to on-device processing. By contrast, the *edge* configuration attains the lowest latency among the evaluated options, but this improvement comes at the cost of substantially increased energy usage driven by continuous wireless data transmission. The *greedy* strategy yields inferior results overall: it incurs higher latency than the *edge* configuration while drawing a similar amount of energy, and thus fails to offer a favorable balance between these two metrics.

Across all device populations, the proposed *LLM* controller functions near the Pareto-optimal frontier of the latency–energy trade-off.

For N=12 devices (Fig. 7.4(a)), the LLM controller reduces latency by more than 40% relative to the local execution baseline while increasing energy consumption only moderately. When the number of devices increases to N=24 (Fig. 7.4(b)), the controller maintains a near-optimal position on the trade-off curve and exhibits lower variance across repeated runs. A similar trend is observed for N=36 devices (Fig. 7.4(c)), where the LLM controller continues to outperform the greedy policy and remains close to edge-level latency while requiring substantially less energy.

Table 7.1: Performance comparison under baseline conditions.

Controller	Latency (ms)	Energy (J)	Deadline Violations	Offload Ratio
Local	440.8 ± 16.2	0.062 ± 0.002	0.771 ± 0.045	0.00
Greedy	263.7 ± 5.7	0.147 ± 0.004	0.423 ± 0.025	0.92
LLM	252.5 ± 4.1	0.125 ± 0.003	0.375 ± 0.027	0.81
Edge	240.1 ± 4.9	0.153 ± 0.005	0.359 ± 0.018	1.00



(a) N=12

(b) N=24

(c) N=36

Figure 7.4: Latency–energy trade-offs for baseline methods.

The comparatively small error bars for the LLM results indicate stable and consistent decision-making. This stability implies that the controller converges to a dependable scheduling strategy that balances computational load distribution and communication overhead across different device populations.

7.5 Performance Under High Workload Conditions

Table 7.2 summerizes system behavior in high-load conditions, where the task arrival rate is doubled to emulate bursty, congested traffic. Under this intensified workload, naive on-device processing is rapidly saturated: devices hit their computational limits, deplete their batteries, and still fall short of meeting tight real-time deadlines. Purely offloading all jobs to the edge and the greedy heuristic baselines manage to reduce latency, but only by funneling a large fraction of requests to the edge infrastructure, which in turn sharply raises communication overhead and energy consumption.

In contrast, the LLM-based controller adopts a more selective and context-aware adaptation mechanism. DT workload predictions and FL synchronization signals are exploited to pinpoint exactly which tasks benefit the most from being offloaded, in terms of latency reduction and deadline satisfaction. This fine-grained policy focuses offloading on latency-critical tasks and keeps others local when advantageous, yielding response times comparable to those of the more aggressive offloading schemes while substantially lowering overall energy usage. These findings suggest that the LLM effectively captures the temporal dynamics and spatial heterogeneity of the workload, and can continuously adjust the offloading rate and the level of device-edge collaboration as conditions evolve.

Furthermore, when the platform is pushed close to its computational and networking limits and the system operates in a regime where congestion effects become pronounced, the proposed framework still consistently meets stringent real-time deadlines with a high success rate. It maintains this stable performance without any additional LLM retraining or heavy hyperparameter reconfiguration, highlighting the robustness, adaptability, and generalization strength of the LLM-guided decision engine in volatile, high-stress operating scenarios.

Table 7.2: Performance comparison under high-load conditions.

Controller	Latency (ms)	Energy (J)	Deadline Violations	Offload Ratio
Local	434.0 ± 7.7	0.061 ± 0.0011	0.749 ± 0.013	0.00
Greedy	245.3 ± 5.4	0.150 ± 0.0027	0.370 ± 0.026	0.95
LLM	253.8 ± 5.3	0.125 ± 0.0009	0.378 ± 0.026	0.78
Edge	241.3 ± 4.0	0.154 ± 0.0030	0.386 ± 0.009	1.00

7.6 Results Under Heterogeneous Device Conditions

Table 7.3 presents the system's performance under heterogeneous device settings, where IoT nodes differ in their computational strength and resource availability. Such diversity adds a further dimension of difficulty, since disparities in processing capabilities and energy budgets render uniform offloading policies inherently inefficient. Purely local execution performs inadequately because it cannot alleviate the bottleneck introduced by weaker devices, leading to slower overall task completion and frequent deadline breaches.

While both full edge offloading and greedy heuristics reduce latency, they incur substantially higher total energy consumption, as even powerful devices offload tasks that they could have handled locally with lower cost.

By contrast, the LLM-assisted controller continuously interprets each node's contextual characteristics including its DT-based performance prediction and the feedback from the federated learning model to adapt and refine offloading decisions in real time.

Through this context-aware policy, high-performance nodes are encouraged to execute tasks locally when beneficial, whereas more demanding or time-critical workloads are selectively routed away from constrained nodes, thereby preserving a favorable trade-off between latency and energy at the network scale.

The observed consistency in both delay and energy profiles indicates that the LLM-based controller effectively generalizes to heterogeneous operating conditions without the need for explicit retraining, per-device tuning, or manual parameter adjustment, highlighting its intrinsic adaptability and robustness in mixed-resource

IoT deployments.

Table 7.3: Performance comparison under heterogeneous device conditions.

Controller	**Latency (ms)**	**Energy (J)**	**Deadline Violations**	**Offload Ratio**
Local	481.0 ± 18.4	0.062 ± 0.0013	0.797 ± 0.016	0.00
Greedy	261.6 ± 4.9	0.145 ± 0.0020	0.413 ± 0.035	0.91
LLM	253.6 ± 2.3	0.129 ± 0.0030	0.383 ± 0.026	0.83
Edge	244.3 ± 6.1	0.153 ± 0.0026	0.382 ± 0.024	1.00

7.7 Performance Under Unstable Wireless Networks

Wireless instability represents one of the most challenging operating environments for IoT deployments, marked by unpredictable latency spikes and variable bandwidth that interfere with reliable task scheduling.

As reported in Table 7.4, the LLM-assisted controller preserves consistently high performance under these volatile conditions, attaining latency that is nearly identical to full edge offloading (276.1 ± 4.4 ms vs. 270.5 ± 9.3 ms) while using about 21% less energy (0.122 ± 0.004 J vs. 0.154 ± 0.0027 J).

This gain is driven by the LLM's contextual reasoning, which fuses DT-based predictions with FL convergence indicators to foresee channel deterioration in advance rather than reacting after the fact.

Rather than opportunistically exploiting every short-lived low-latency interval, as the greedy baseline does, the LLM issues offloading decisions only when both link stability and energy state jointly support a beneficial transfer.

By doing so, it avoids redundant or low-yield transmissions, thereby conserving energy, reducing queuing and retransmissions, and improving overall reliability in highly dynamic conditions. While full edge offloading can still offer marginally better latency, its performance is tightly coupled to the availability of a stable wireless link, and it incurs a higher energy budget, which is problematic for battery-constrained

Table 7.4: Performance comparison under unstable wireless conditions.

Controller	**Latency (ms)**	**Energy (J)**	**Deadline Violations**	**Offload Ratio**
Local	433.0 ± 11.8	0.061 ± 0.0017	0.753 ± 0.024	0.00
Greedy	293.5 ± 6.6	0.140 ± 0.0020	0.511 ± 0.022	0.83
LLM	276.1 ± 4.4	0.122 ± 0.0040	0.464 ± 0.016	0.76
Edge	270.5 ± 9.3	0.154 ± 0.0027	0.472 ± 0.028	1.00

devices.

Taken together, these results indicate that the LLM-based controller achieves a more favorable balance between responsiveness and energy consumption, exhibiting robustness to random wireless fluctuations through proactive, context-aware scheduling that adapts to both current and predicted network states.

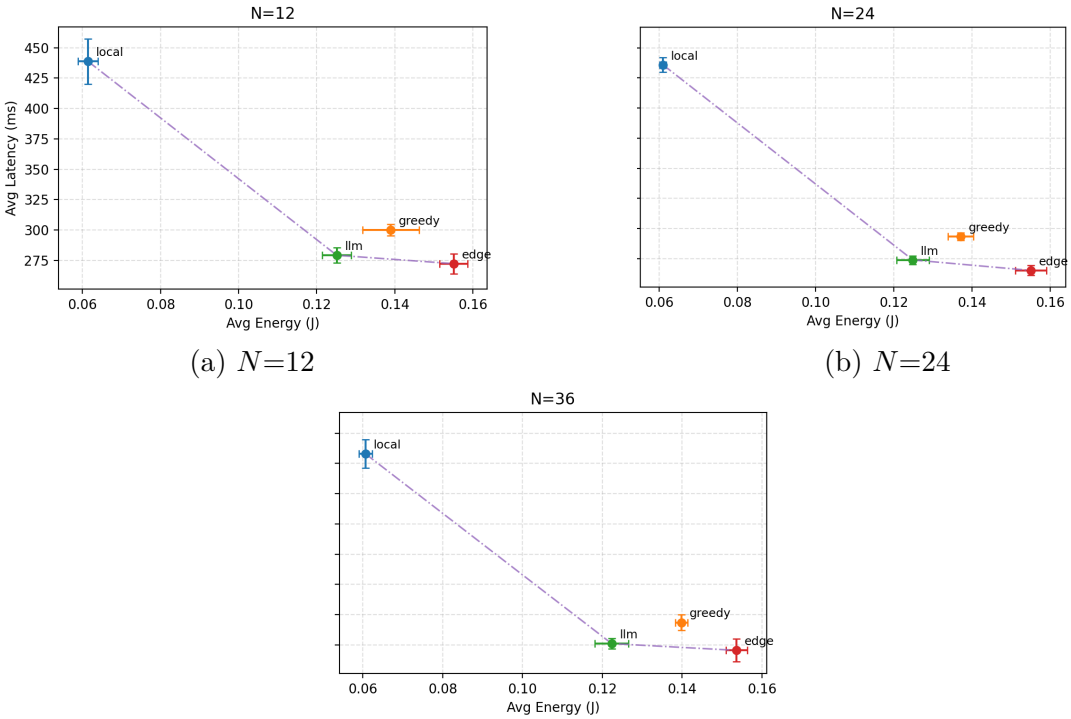
Figure 7.5 presents the latency–energy trade-offs under unstable wireless conditions across different numbers of devices. The patterns are consistent with the baseline case: *local* execution yields the lowest energy consumption but the highest latency, whereas *edge* offloading lowers latency at the cost of higher energy usage. The *greedy* strategy continues to be suboptimal, incurring higher latency and energy than the *LLM*-based controller. In comparison, the proposed *LLM* solution stays close to the optimal trade-off curve.

For N=12 (Fig. 7.5(a)), the *LLM* already cuts latency relative to *local*, while consuming less energy than full *edge* offloading.

When the network grows to N=24 (Fig. 7.5(b)), the performance gap between *LLM* and *edge* shrinks, showing that the learned policy generalizes well as contention rises.

At N=36 (Fig. 7.5(c)), the benefits of *LLM* become particularly pronounced: it attains an average latency of 276.1 ± 4.4 **ms** with an energy expenditure of just 0.122 ± 0.004 **J**, clearly below that of *edge* offloading (0.154 ± 0.0027 **J**).

Even under wireless variability, the *LLM* controller meets deadlines at a rate comparable to *edge* (0.464 ± 0.016 vs. 0.472 ± 0.028), highlighting its stability and efficiency in challenging conditions.



(a) N=12 (b) N=24

(c) N=36

Figure 7.5: Latency-energy trade-offs under unstable wireless conditions.

7.8 Explainability and Trust Analysis

Beyond examining system-level metrics such as energy efficiency and latency, it is also crucial to evaluate how interpretable and reliable the learned decision policies are. In intelligent IoT scenarios where autonomous controllers choose how tasks are executed, explainability tools help verify that decisions are transparent and consistent with the intended system behavior. This section, therefore, explores the explainability aspects of the proposed framework by studying feature importance and saliency alignment within the learned policy.

Instead of focusing on the internal architecture of the RL agent, the analysis focuses on the behavior of the agent over several training episodes and on how different state variables shape its decisions. In particular, it examines exploration patterns, battery sustainability, offloading behavior, and gradient-based feature attribution.

7.8.1 Exploration Rate Dynamics

Figure 7.6 illustrates the evolution of the exploration parameter ϵ-greedy across training episodes. At the beginning of training, the exploration rate is intentionally high to encourage extensive sampling of the action space. This enables the agent to experience a diverse set of system states, including varying CPU loads, network latencies, and battery conditions. The "greedy" part refers to the default behavior, most of the time, the agent picks the action it currently believes is best (the one with the highest estimated reward).

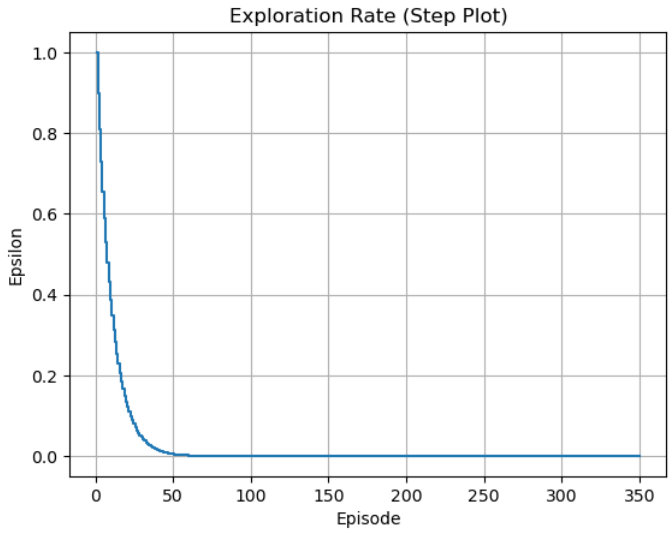


Figure 7.6 Decay of the exploration rate in training episodes.

A rapid decrease is seen during the first 50 episodes, after which the exploration rate levels off close to zero. This pattern reflects a shift from exploration to exploitation, enabling the controller to reliably use the learned offloading policy once enough information about the environment has been gathered. Such quick convergence is especially important in IoT systems, where constrained computational resources make extended training impractical.

7.8.2 Battery Retention and Energy Stability

The sustainability of the learned policy is assessed by tracking how battery levels change over the training episodes. Figure 7.7 shows the average end-of-episode battery level across all nodes.

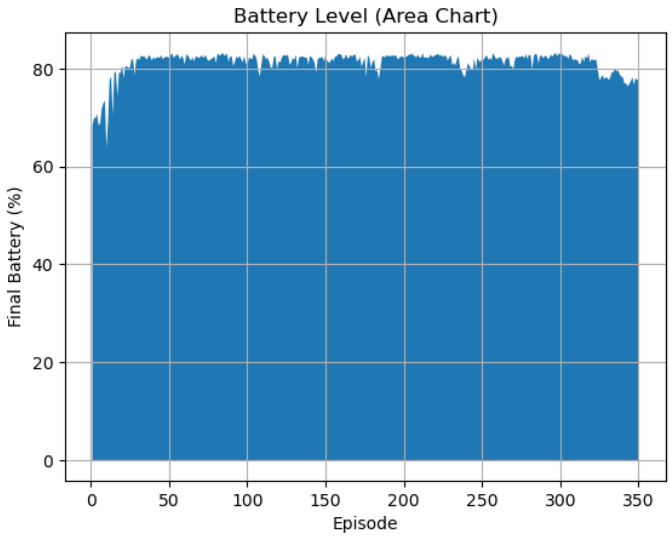


Figure 7.7 Battery level evolution across 350 training episodes.

The results indicate that battery charge remains consistently high, typically between 82% and 83%. Small variations occur in the early training phase due to exploratory actions, but the system rapidly stabilizes once the learned policy starts prioritizing energy-efficient choices.

These results verify that the reward scheme effectively drives the controller to trade off computational performance against energy conservation. Episodes with very low battery usage usually arise when the controller locates advantageous offloading options while avoiding superfluous communication overhead.

7.8.3 Offloading Behavior Distribution

To gain deeper insight into the behavioral properties of the learned policy, we examine how offloading actions are distributed across episodes. Figure 7.8 shows the empirical distribution of offload counts observed during training.

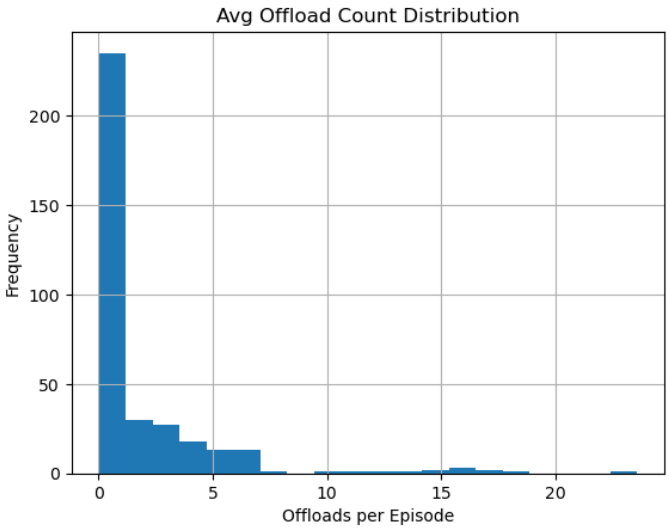


Figure 7.8 Distribution of average offloading actions per episode.

The histogram shows that in most episodes, fewer than five offloading actions are taken. Still, the distribution has a long tail, with some episodes involving more than fifteen offloads. These high-offload episodes typically arise when local computational demand rises or when network latency spikes, making offloading more beneficial.

This heavy-tailed pattern suggests that the learned policy generally follows a conservative offloading strategy under normal conditions, yet remains capable of aggressively offloading tasks during atypical situations. Such adaptability is essential in real-world IoT settings, where resource availability and workload levels are constantly changing.

7.8.4 Feature Relevance and Saliency Alignment

To gain a clearer view of how the controller makes its decisions, we apply a gradient-based saliency method. This explainability approach quantifies the influence of each input feature on the predicted Q-values that drive offloading choices.

Table 7.5 reports the average absolute gradient values for each state variable over all evaluated decisions.

The saliency analysis shows that network latency and CPU utilization are the primary drivers of offloading decisions. This aligns with the reward function's design, which penalizes both high latency and heavy computational load.

While battery level has a comparatively smaller gradient magnitude, its role

Table 7.5: Average gradient magnitudes derived from saliency analysis.

Feature	**Avg** $\lvert\nabla Q\rvert$
Network Latency (ℓ)	0.412
CPU Load (c)	0.377
Average CPU ($\bar{c}$)	0.305
Battery Level (b)	0.286

among the influential features confirms that the controller still accounts for energy usage when choosing actions. Put differently, immediate performance constraints dominate the decisions, but long-term energy efficiency remains implicitly embedded in the learned policy.

This consistency between feature importance and design goals increases trust in the controller's behavior. By confirming that the agent emphasizes relevant operational signals, the explainability analysis indicates that the learned policy is not only effective but also interpretable.

7.8.5 Implications for Trustworthy IoT Decision Intelligence

Overall, the explainability analysis yields several key insights into the behavior of the proposed framework:

- **Efficient learning:** The ϵ-greedy exploration strategy converges quickly, enabling the controller to stabilize its decision policy early in the training process.
- **Energy stability:** Battery levels consistently remain above 80%, indicating that the learned policy effectively balances computational performance with energy conservation.
- **Adaptive offloading behavior:** The pattern of offloading actions shows that the controller continuously adjusts to changing CPU load and network conditions.
- **Interpretability:** Gradient-based saliency analysis indicates that the controller focuses on operationally relevant features, enhancing transparency and supporting confidence in automated decisions.

Together, these results underscore the value of embedding explainability mechanisms within intelligent IoT control frameworks. By coupling reinforcement learning with interpretable feature attribution, the proposed system not only provides efficient task scheduling but also exposes the rationale behind its decisions. This transparency is crucial to establishing reliable autonomous systems in resource-constrained IoT settings.

The final chapter summarizes the main findings of this research and reflects on the contributions of the proposed framework for intelligent task offloading in IoT environments. It also discusses the limitations of the current study and outlines potential directions for future research in adaptive, explainable, and energy-aware distributed systems.

Chapter 8

Conclusions and Future Work

This dissertation examined how to design intelligent and reliable task offloading strategies for distributed IoT systems that operate under changing resource conditions. Contemporary IoT settings must cope with several issues at once, such as constrained energy resources, variable network performance, diverse device capabilities, and growing privacy demands. Conventional reactive offloading methods frequently fail to deliver consistent performance in these environments. To address these challenges, this thesis proposed a unified framework that combines FL, DT-based MARL control strategies, LLMs, and XAI methods to support proactive and interpretable task offloading decisions. Comprehensive simulation experiments and analytical evaluations demonstrated that the framework can successfully trade off energy consumption, computational performance, and decision explainability in distributed IoT environments.

The findings in the preceding chapters demonstrate that integrating predictive system modeling, decentralized collaborative learning, and explainable decision processes allows IoT networks to function more efficiently and reliably under diverse operating conditions. The proposed framework thus offers both conceptual and methodological progress toward intelligent edge computing systems that support adaptive and trustworthy decision-making.

8.1 Summary of Scientific Contributions

The scientific contributions of this dissertation were outlined in the opening chapter and guided the research methodology and experimental evaluation used throughout. Rather than restating those contributions, this section discusses how the empirical findings support the proposed research direction and demonstrate the effectiveness of the developed framework.

The experimental results show that combining predictive DT and FL with adaptive learning strategies markedly enhances the stability and responsiveness of task offloading policies in dynamic IoT settings. By using short-term predictions of system variables, including battery state, CPU usage, and communication delay, the framework supports proactive decisions instead of relying solely on reactive scheduling. In a range of scenarios covering stable connectivity, high-load conditions, heterogeneous device resources, and varying wireless quality, the predictive module consistently improved resource awareness and led to more stable task allocation.

The findings further indicate that federated training methods enable collaborative learning in distributed IoT networks without needing centralized access to raw device data. Through the federated learning framework, individual nodes gained from shared policy updates while keeping local data private. Experimental results demonstrated that decentralized learning still converges to a stable global policy, showing that distributed model updates can preserve learning performance while lowering privacy risks and communication overhead.

Another key result of the experimental analysis is the confirmation that explainability methods enhance the transparency of autonomous decision-making. Gradient-based saliency analysis showed that the learned policy reliably focuses on operationally relevant features, such as network latency and CPU usage, when selecting offloading actions. This match between feature importance and system objectives indicates that the controller acts in line with the intended optimization goals, offering interpretable insight into how its decisions are formed.

Overall, the experiments show that adding LLM-driven reasoning to the offloading pipeline yields an effective decision-support mechanism that can adjust to heterogeneous, rapidly changing environments. By fusing predictive signals from DTs

with learning feedback from federated models, the LLM-based controller produced context-aware scheduling choices that remained robust under a wide range of operating conditions.

Finally, these findings show that the proposed framework effectively integrates predictive modeling, decentralized learning, and interpretable decision-making to enable proactive task management in IoT systems. The experimental results thus offer strong support that the research contributions presented in this dissertation are both practically feasible and scientifically verified.

8.1.1 Validation of Research Hypotheses

The experimental evaluation conducted throughout this dissertation confirms the validity of the research hypotheses introduced in earlier chapters.

Hypothesis 1 – Digital Twin–Enhanced Learning. The findings show that integrating DT predictions into the learning process markedly enhances the stability and agility of offloading decisions. By using forecasts of battery status, CPU load, and network latency, the system can foresee impending resource limitations and adjust scheduling policies in advance. Experiments over various scenarios confirm that this predictive capability produces more stable policies and better energy preservation than purely reactive methods.

Hypothesis 2 – Federated MARL and Privacy Preservation. The federated training framework effectively supports distributed learning without transmitting raw sensor data. Experiments indicate that agents updated through federated aggregation converge to a shared global policy with performance similar to centralized methods. Meanwhile, the decentralized procedure greatly enhances privacy and lowers communication costs, demonstrating that federated learning is practical for large-scale IoT settings.

Hypothesis 3 – Explainability and Trust. The use of gradient-based explainability methods offers clear visibility into how the controller makes its decisions. Saliency analysis shows that critical system variables, such as network latency and CPU load, have a strong impact on offloading choices. This match between feature relevance and system design goals demonstrates that the controller operates in

line with the intended optimization objectives, enhancing the interpretability of and confidence in autonomous decision-making.

Overall, the empirical findings support the hypothesis of this dissertation: integrating predictive DTs, federated distributed learning, and explainable decision mechanisms markedly enhances the robustness, efficiency, and trustworthiness of task offloading in IoT networks.

8.2 Limitations of the Study

Although the proposed framework demonstrates promising results, several limitations must be considered.

First, more extensive research should be conducted in real-world scenarios. Although the present study models network latency, computational load, and energy consumption in a realistic manner, broader real-world IoT deployments may involve additional challenges, such as hardware heterogeneity, changing environmental conditions, and possible communication disruptions.

Second, the present experiments target medium-sized IoT networks with a limited node count. While the framework is intended to scale to larger deployments, further experiments are needed to assess its performance in very large distributed environments with thousands of devices or more.

Third, the explainability analysis relies mainly on gradient-based saliency methods. Although these techniques offer useful information about feature importance, they reflect only one type of explainability approach. More advanced interpretability methods could yield a deeper understanding of the decision-making process and policy dynamics.

Finally, the LLM-based decision support system was assessed in a controlled setting using predefined prompts and system signals. Additional studies are needed to examine how LLM reasoning performs under highly varied real-world workloads and fluctuating network conditions.

8.3 Future Research Directions

The results of this dissertation suggest several promising directions for future work.

A key direction is the incorporation of hardware-in-the-loop experiments. By linking the proposed framework to actual IoT devices and edge computing platforms, subsequent research could assess system performance under realistic hardware limitations and network conditions. This type of validation would yield a more nuanced understanding of the practical viability of deploying proactive offloading strategies.

Another promising direction concerns deployment in real-world IoT settings. Using the proposed framework in areas such as smart cities, industrial monitoring, and intelligent transportation networks would enable assessment under varied operating conditions and across heterogeneous device ecosystems.

An important direction for future work is to create more rigorous techniques for interpreting and explaining how distributed intelligent systems make decisions. In this regard, counterfactual explanations are especially useful because they indicate the smallest changes needed for the system to reach a different outcome. Moreover, model-agnostic interpretability tools such as SHAP [96, 97, 98] and LIME [99, 100] can generate human-readable explanations for individual decisions without relying on the internal structure of the model itself. Combining these techniques with MARL agents and LLM-based controllers in IoT settings would extend analysis beyond merely describing system behavior to achieving a deeper insight into the underlying driving factors. Such transparency is crucial for real-world applications, where trust, accountability, and the ability to identify and correct problematic behavior are vital. Another promising direction is to examine how well the proposed framework holds up in adversarial settings, such as when sensor data or communication channels are deliberately compromised. Understanding how the system reacts to erroneous or maliciously crafted inputs is essential for its eventual deployment in real-world conditions.

Integrating energy harvesting is another compelling research direction, where the framework is extended to include devices that recharge from environmental sources like solar or kinetic energy, adding an extra layer of uncertainty that the predictive DT must capture.

By pursuing these research directions, future studies can further reinforce the foundations of explainable and energy-efficient distributed intelligence, thereby bringing autonomous IoT systems closer to reliable real-world deployment.

Curriculum Vitae

Klea Çapari Elmazi was born in Tirana, Albania, on 15 May 1997. After completing her primary and secondary education, she enrolled in the undergraduate study program in Information and Communication Technology at the University of Tirana in 2015, where she obtained her Bachelor of Science degree in 2018.

In 2018, she continued her education by enrolling in a Master of Science program in Computer Engineering, Network and Cybersecurity profile at the Canadian Institute of Technology, Albania, which she completed in 2020. During her studies, she developed a strong interest in secure systems, distributed architectures, and data-driven technologies.

Following her graduate studies, she began her academic career as a lecturer at the Canadian Institute of Technology in Tirana, where she taught courses such as Python Programming, Software Architecture, and Computer Applications. She later continued her academic engagement at Luarasi University, contributing to both undergraduate and graduate teaching in areas including Java Programming, Business Intelligence, and Big Data Management. Since 2025, she has been serving as a lecturer in the Department of Software Engineering at the Canadian Institute of Technology, where she teaches courses in Artificial Intelligence, Machine Learning, Data Analytics, and Data Visualization, while actively participating in research and academic project development.

In 2022, she enrolled in a postgraduate doctoral study in the field of technical sciences, specifically computer science, at the University of Rijeka, Faculty of Engineering. Her research focuses on federated learning, edge intelligence, distributed systems, and energy-effective IoT architectures.

During her doctoral studies, she has authored and co-authored numerous sci-

entific publications in international journals and conferences, including works published in Elsevier, Springer, and IEEE. Her research contributions primarily address intelligent task offloading, digital twin-driven systems, explainable artificial intelligence, and federated multi-agent learning in IoT environments.

She has also participated in research and innovation projects related to smart city systems, particularly focusing on intelligent transportation and IoT-based traffic optimization, supported by national research initiatives from Croatia and Albania.

List of Publications

Journal Papers

[1] Elmazi, K.; Elmazi, D.; Lerga, J. Digital Twin-Driven Federated Learning and Reinforcement Learning-Based Offloading for Energy-Efficient Distributed Intelligence in IoT Networks. *Internet of Things*, Elsevier, Vol. 32, Article 101640, 2025, IF: 7.6, CiteScore: 12.4.doi:10.1016/j.iot.2025.101640.

[2] Elmazi, K.; Elmazi, D.; Lerga, J. XDT-FMARL: An Explainable Federated Multi-Agent Reinforcement Learning Framework for Energy-Efficient IoT Task Offloading. *International Journal of Web and Grid Services*, Inderscience Publishers, 2025, Vol. 21, Issues 3–4, pp. 398–423, IF:1.4, CiteScore: 3.2.doi:10.1504/IJWGS.2025.10073957.

[3] Elmazi, K.; Elmazi, D.; Lerga, J. Proactive Context-Aware Task Offloading in Digital Twin–Driven Federated IoT Systems with Large Language Models. *Internet of Things*, Elsevier, 2025, Article 101826,IF: 7.6, CiteScore: 12.4.doi:10.1016/j.iot.2025.101826.

[4] Çapari, K.; Elmazi, D.; Prieditis, M. Efficiency Performance Evaluation on Multi-user Web Application Platforms in Cloud Computing. *International Journal of Innovative Technology and Interdisciplinary Sciences* (IJITIS), 2022, Vol. 5, No. 3, pp. 1014–1032, IF: 1.2. doi:10.15157/IJITIS.2022.5.3.1014-1032.

Conference Papers

[5] Elmazi, K.; Elmazi, D.; Lerga, J. Federated Virtual Sensors for IoT: Applying Machine Learning Algorithms Through Federated Averaging and Distributed Intelligence. In Advances on Broad-Band Wireless Computing, Communication and

Applications: Proceedings of the 19th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA-2024); *Lecture Notes on Data Engineering and Communications Technologies*, Vol. 231; Springer: Cham, Switzerland, 2024; pp. 141–151. doi:10.1007/978-3-031-76452-3_14

[6] Elmazi, K.; Elmazi, D.; Lerga, J. Advancing Federated Virtual Sensors for IoT with Privacy Safeguards and Robustness to Adversarial Threats. In Advances in Internet, Data and Web Technologies: Proceedings of the 13th International Conference on Emerging Internet, Data and Web Technologies (EIDWT-2025); *Lecture Notes on Data Engineering and Communications Technologies*, Vol. 243; Springer: Cham, Switzerland, 2025; pp. 159–172. doi:10.1007/978-3-031-86149-9_16.

[7] Elmazi, K.; Elmazi, D.; Lerga, J. Adaptive Offloading in Federated Virtual Sensors for Efficient Resource Management in IoT Systems. In Advanced Information Networking and Applications: Proceedings of the 39th International Conference on Advanced Information Networking and Applications (AINA-2025), Volume 2; *Lecture Notes on Data Engineering and Communications Technologies*, Vol. 246; Springer: Cham, Switzerland, 2025; pp. 390–404. doi:10.1007/978-3-031-87766-7.

[8] Elmazi, K.; Elmazi, D.; Musta, E.; Mehmeti, F.; Hidri, F. An Intelligent Transportation Systems-Based Machine Learning-Enhanced Traffic Prediction Model using Time Series Analysis and Regression Techniques. In Proceedings of the 2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA 2024); IEEE: Craiova, Romania; pp. 1–6.doi:10.1109/INISTA62901.2024.10683864

[9] Çapari, K.; Elmazi, D. Evaluating Support Vector Machine and Logistic Regression: A Machine Learning-Based Statistical Study for Breast Cancer Diagnosis. In Proceedings of the 2024 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA 2024); IEEE: Tirana, Albania, 18–19 December 2024. doi:10.1109/CoNTESA64738.2024.10891283

[10] Musta, E.; Elmazi, D.; Elmazi, K.; Mehmeti, F.; Hidri, F. Testing Intelligent Traffic Control Solutions efficiency in reducing traffic and pollution in Tirana. In Proceedings of the International Workshop on Quantum & Biomedical Applications,

Technologies, and Sensors (Q-BATS 2024); IEEE: Durrës, Albania, 10–11 October 2024; pp. 121–126. doi:10.1109/Q-BATS64589.2024.10873945

[11] Çapari, K.; Baraj, A. Level-based Information Security Analysis and Implementation. In Proceedings of the International Conference on Digital Economy and Recent Technology Trends; Canadian Institute of Technology (CIT): Tirana, Albania, 23 April 2021.

[12] Elmazi, D.; Karras, D. A.; Alkholidi, A. G.; Çapari, K. Cybersecurity and Privacy Attacks Detection in IoT Networks with Improved Data Engineering and Machine Learning Methods. In Proceedings of the IEEE Ninth International Conference on Big Data Computing Service and Applications (BigDataService 2023); IEEE: Athens, Greece,pp. 223–228. doi:10.1109/BigDataService58306.2023.10234007

[13] Elmazi, K.; Elmazi, D.; Fetaji, B. Evaluating Spectral Clustering for Intrusion Detection: A Performance-Based Approach. In *Book of Abstracts*

Bibliography

[1] Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A Survey. *Computer Networks* **54**(15), 2787–2805. Elsevier (2010).doi:10.1016/j.comnet.2010.05.010

[2] H. Tran-Dang and D.-S. Kim, "Digital Twin-Empowered Intelligent Computation Offloading for Edge Computing in the Era of 5G and Beyond: A State-of-the-Art Survey," *ICT Express*, vol. 11, no. 1, pp. 167–180, 2025. doi:10.1016/j.icte.2025.01.002

[3] Zhang, C.: Intelligent Internet of Things Service Based on Artificial Intelligence Technology. In: *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pp. 731–734. IEEE (2021).doi:10.1109/ICBAIE52039.2021.9390061

[4] Rose, K., Eldridge, S., Chapin, L.: The Internet of Things: An Overview. Internet Society, Reston, VA (2015). `https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf`

[5] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020. doi:10.1109/COMST.2020.3009103

[6] O. Aouedi, T. H. Vu, A. Sacco, D. C. Nguyen, K. Piamrat, G. Guillouard, and A. Bhardwaj, "A Survey on Intelligent Internet of Things: Applications, Security, Privacy, and Future Directions," *IEEE Communications Surveys and Tutorials*, vol. 26, no. 4, pp. 2417–2464, 2024. doi:10.1109/COMST.2024.3430368

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. doi:10.1109/JIOT.2016.2579198

[8] M. Mahbub and R. M. Shubair, “Contemporary Advances in Multi-Access Edge Computing: A Survey of Fundamentals, Architecture, Technologies, Deployment Cases, Security, Challenges, and Directions,” *Journal of Network and Computer Applications*, vol. 219, p. 103726, 2023. doi:10.1016/j.jnca.2023.103726

[9] Liu, B., Lv, N., Guo, Y., Li, Y.: Recent advances on federated learning: A systematic survey. arXiv preprint arXiv:2301.01299 (2023). doi:10.48550/arXiv.2301.01299

[10] Khan, L.U., Saad, W., Han, Z., Hossain, E., Hong, C.S.: Federated learning for internet of things: Recent advances, taxonomy, and open challenges. IEEE Communications Surveys Tutorials **23**(3), 1759–1799 (2021). doi:10.1109/COMST.2021.3090430

[11] Madria, S., Kumar, V., Dalvi, R.: Sensor cloud: A cloud of virtual sensors. IEEE Software **31**(2), 70–77 (2013). doi:10.1109/MS.2013.141

[12] Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492. doi:10.48550/arXiv.1610.05492

[13] Thonglek, K., Takahashi, K., Ichikawa, K., Iida, H., Nakasan, C.: Federated learning of neural network models with heterogeneous structures. In: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 735–740. IEEE (2020) doi:10.1109/ICMLA51294.2020.00120

[14] Moayad Aloqaily, Ismaeel Al Ridhawi, and Salil Kanhere, "Reinforcing Industry 4.0 With Digital Twins and Blockchain-Assisted Federated Learning," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3504-3516, 2023. doi:10.1109/JSAC.2023.3310068

[15] Baek, M.-S., Jung, E., Park, Y.S., Lee, Y.-T.: Federated Digital Twin Implementation Methodology to Build a Large-Scale Digital Twin System. In: *2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–2. IEEE (2024). doi:10.1109/BMSB62888.2024.10608284

[16] Baek, M.-S.: Digital Twin Federation and Data Validation Method. In: *2022 27th Asia Pacific Conference on Communications (APCC)*, pp. 445–446. IEEE (2022). doi:10.1109/APCC55198.2022.9943622

[17] Hongwei Liu, Nan Tian, Dong-An Song, and Li Zhang, "Digital Twin-Enabled Multi-Service Task Offloading in Vehicular Edge Computing Using Soft Actor-Critic," *Electronics*, vol. 14, no. 4, p. 686, 2025, publisher: MDPI. doi:10.3390/electronics14040686

[18] A. Mohamed, K. J. Abdelqader, and K. Shaalan, "Explainable Artificial Intelligence: A Systematic Review of Progress and Challenges," *Intelligent Systems with Applications*, vol. 28, p. 200595, 2025. doi:10.1016/j.iswa.2025.200595

[19] A. Karras, A. Giannaros, N. Amasiadi, and C. Karras, "Next-Gen Explainable AI (XAI) for Federated and Distributed Internet of Things Systems: A State-of-the-Art Survey," *Future Internet*, vol. 18, no. 2, p. 83, 2026. doi:10.3390/fi18020083

[20] Klea Elmazi, Donald Elmazi, and Jonatan Lerga, "Digital Twin-Driven Federated Learning and Reinforcement Learning-Based Offloading for Energy-Efficient Distributed Intelligence in IoT Networks," *Internet of Things*, p. 101640, 2025, publisher: Elsevier. doi:10.1016/j.iot.2025.101640

[21] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," in *Handbook of Reinforcement Learning and Control*, Springer, 2021, pp. 321–384, doi:10.1007/978-3-030-60990-0_12

[22] A. Rafique and B. D. Marsden, "Automated LLM Deployment and Evaluation: A Cloud-Native Approach Using LLM-as-a-Judge," in *Proc. IEEE 18th*

International Conference on Cloud Computing (CLOUD), pp. 448–450, 2025. doi:10.1109/CLOUD67622.2025.00053

[23] S. Yeo, S. Lee, B. Choi, and S. Oh, "Integrate Multi-Agent Simulation Environment and Multi-Agent Reinforcement Learning (MARL) for Real-World Scenario," in *Proc. 2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 523–525, 2020. doi:10.1109/ICTC49870.2020.9289369

[24] Y. Zhou, M. Zhou, and F. Yu, "MARL-Based Multi-Hop Task Offloading for High-Density VEC Networks," in *Proc. IEEE 4th International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT)*, pp. 1–5, 2025. doi:10.1109/CCPQT66408.2025.11383366

[25] P. R. X. do Carmo, D. de Freitas Bezerra, A. T. Oliveira Filho, S. Campelo, P. Pedrosa, and J. Kelner, "Living on the Edge: A Survey of Digital Twin-Assisted Task Offloading in Safety-Critical Environments," *Journal of Network and Computer Applications*, vol. 232, p. 104033, 2024. doi:10.1016/j.jnca.2024.104024

[26] J. Chen, Z. Cai, W. Chen, W. Wang, Z. Zheng, and P. S. Yu, "A Federated Adaptive Large Language Model Fine-Tuning Framework for Software Development," *IEEE Transactions on Services Computing*, vol. 19, no. 1, pp. 32–43, 2026. doi:10.1109/TSC.2025.3623626

[27] A. A. Ismail, N. E. Khalifa, and R. A. El-Khoribi, "A Survey on Resource Scheduling Approaches in Multi-Access Edge Computing Environment: A Deep Reinforcement Learning Study," *Cluster Computing*, vol. 28, article 184, 2025. doi:10.1007/s10586-024-04893-7

[28] Brdnik, S., Šumak, B.: Current Trends, Challenges and Techniques in XAI Field: A Tertiary Study of XAI Research. In: *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pp. 2032–2038. IEEE (2024). doi:10.1109/MIPRO60963.2024.10569528

[29] P. Mach and Z. Běcvář, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017. doi:10.1109/COMST.2017.2682318

[30] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. doi:10.1109/COMST.2017.2745201

[31] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation Offloading in Mobile Edge Computing Networks: A Survey," *Journal of Network and Computer Applications*, vol. 202, p. 103366, 2022. doi:10.1016/j.jnca.2022.103366

[32] Shi Dong, Junxiao Tang, Khushnood Abbas, Ruizhe Hou, Joarder Kamruzzaman, Leszek Rutkowski, and Rajkumar Buyya, "Task Offloading Strategies for Mobile Edge Computing: A Survey," *Computer Networks*, Elsevier, 2024. doi:10.1016/j.comnet.2024.110791

[33] X. Zhang and S. Debroy, "Resource Management in Mobile Edge Computing: A Survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–36, 2023. doi:10.1145/3589639

[34] Z. Luo and X. Dai, "Reinforcement Learning-Based Computation Offloading in Edge Computing: Principles, Methods, Challenges," *Alexandria Engineering Journal*, vol. 104, pp. 394–405, 2024. doi:10.1016/j.aej.2024.07.049

[35] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.doi:10.1109/JSAC.2016.2611964

[36] J. Pournazari, A. Ullah, A. Al-Dubai, and X. Liu, "Computation Offloading in the Edge–Cloud Computing Continuum," *Cluster Computing*, 2025. doi:10.1007/s10586-025-05577-6

[37] J. Zhang, Q. Cui, X. Zhang, K.-C. Chen, and P. Zhang, "Event-triggered Online Proactive Network Association to Mobile Edge Computing for IoT," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019. doi:10.48550/arXiv.1905.01097

[38] L. Zhang, D. A. Song, H. Zhang, N. Tian, Z. Zhuang, D. Niyato, and Z. Han, "Edge-driven industrial computing power networks: Digital twin-empowered service provisioning by hybrid soft actor-critic," *IEEE Transactions on Vehicular Technology*, vol. 74, 2025, doi:10.1109/TVT.2025.3525704

[39] H. Tran-Dang and D.-S. Kim, "Digital twin-empowered intelligent computation offloading for edge computing in the era of 5G and beyond: A state-of-the-art survey," *ICT Express*, vol. 11, no. 1, pp. 167–180, Feb. 2025, doi:10.1016/j.icte.2025.01.002

[40] N. Rasouli, C. Klein, and E. Elmroth, "Resource management for mission-critical applications in edge computing: Systematic review on recent research and open issues," *ACM Computing Surveys*, vol. 58, no. 3, art. 71, Sept. 2025, doi:10.1145/3762181

[41] T. Deng, Z. Yu, and D. Yuan, "Task offloading optimization in mobile edge computing under uncertain processing cycles and intermittent communications," *Computer Networks*, 2024, doi:10.1016/j.comnet.2024.110359.

[42] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020, doi:10.1109/MSP.2020.2975749

[43] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated Learning for Internet of Things: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.doi:10.1109/COMST.2021.3075439

[44] M. Aggarwal, V. Khullar, S. Rani, T. A. Prola, S. B. Bhattacharjee, S. M. Shawon, and N. Goyal, "Federated Learning on Internet of

Things: Extensive and Systematic Review," *Computers, Materials & Continua*, vol. 79, no. 2, pp. 1795–1834, 2024, doi: 10.32604/cmc.2024.049846. doi:10.32604/cmc.2024.049846

[45] M. H. Alsharif, R. Kannadasan, W. Wei, K. S. Nisar, and A.-H. Abdel-Aty, "A contemporary survey of recent advances in federated learning: Taxonomies, applications, and challenges," *Internet of Things* (Elsevier), vol. 27, Art. 101251, 2024, doi: 10.1016/j.iot.2024.101251.doi:10.1016/j.iot.2024.101251

[46] B. Liu, N. Lv, Y. Guo, and Y. Li, "Recent advances on federated learning: A systematic survey," *Neurocomputing* (Elsevier), vol. 597, Art. 128019, 2024, doi:10.1016/j.neucom.2024.128019.

[47] J. Ayeelyan *et al.*, "Federated learning design and functional models: survey," *Artificial Intelligence Review*, vol. 57, 2024, doi:10.1007/s10462-024-10969-y.

[48] S. K. Jagatheesaperumal *et al.*, "Enabling Trustworthy Federated Learning in Industrial IoT: Bridging the Gap Between Interpretability and Robustness," *IEEE Internet of Things Magazine*, 2024. doi:10.1109/IOTM.001.2300274

[49] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad, H. Venkataraman, R. Trestian, and H. X. Nguyen, "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022, doi:10.1109/COMST.2022.3208773.

[50] H. Xu, J. Wu, Q. Pan, X. Guan, and M. Guizani, "A Survey on Digital Twin for Industrial Internet of Things: Applications, Technologies and Tools," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2569–2598, 2023, doi:10.1109/COMST.2023.3297395.

[51] A. Hakiri, A. Gokhale, S. B. Yahia, and N. Mellouli, "A Comprehensive Survey on Digital Twin for Future Networks and Emerging Internet of Things Industry," *Computer Networks*, vol. 244, p. 110350, 2024. doi:10.1016/j.comnet.2024.110350

[52] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Y. Zeng, B. Ye, and X. Jia, "Digital Twin-Enabled Service Provisioning in Edge Computing via Continual Learning," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7335–7350, 2024, doi:10.1109/TMC.2023.3332668.

[53] Y. Zhang, J. Hu, and G. Min, "Digital Twin-Driven Intelligent Task Offloading for Collaborative Mobile Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3034–3045, 2023, doi:10.1109/JSAC.2023.3310058.

[54] H. Tran-Dang and D.-S. Kim, "Digital Twin-Empowered Intelligent Computation Offloading for Edge Computing in the Era of 5G and Beyond: A State-of-the-Art Survey," *ICT Express*, vol. 11, no. 1, pp. 167–180, 2025, doi:10.1016/j.icte.2025.01.002.

[55] D. Hortelano, I. de Miguel, R. J. Barroso, J. C. Aguado, N. Merayo, L. Ruiz, A. Asensio, X. Masip-Bruin, P. Fernández, and R. M. Lorenzo, "A comprehensive survey on reinforcement-learning-based computation offloading techniques in Edge Computing Systems," *Journal of Network and Computer Applications*, vol. 216, p. 103669, 2023, doi:10.1016/j.jnca.2023.103669.

[56] K. Elmazi, D. Elmazi, and J. Lerga, "Proactive Context-Aware Task Offloading in Digital Twin-Driven Federated IoT Systems with Large Language Models," *Internet of Things*, Art. no. 101826, 2025. doi:10.1016/j.iot.2025.101826

[57] Z. Zabihi *et al.*, "Reinforcement Learning Methods for Computation Offloading: A Systematic Review," *ACM Computing Surveys*, 2023, doi:10.1145/3603703.

[58] B. Xie and H. Cui, "Deep reinforcement learning-based dynamical task offloading for mobile edge computing," *The Journal of Supercomputing*, published Oct. 19, 2024 (Vol. 81, Art. 35, 2025). doi:10.1007/s11227-024-06603-x

[59] Y. Fan *et al.*, "A deep reinforcement approach for computation offloading in mobile edge computing dynamic networks," *EURASIP Journal on Advances in Signal Processing*, 2024, Art. 01142-2. doi:10.1186/s13634-024-01142-2

[60] S. Yao, M. Wang, J. Ren, T. Xia, W. Wang, K. Xu, M. Xu, and H. Zhang, "Multi-Agent Reinforcement Learning for Task Offloading in Crowd-Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 24, pp. 9289–9302, 2025. doi:10.1109/TMC.2025.3531793

[61] J. Xiong, P. Guo, Y. Wang, X. Meng, J. Zhang, L. Qian, and Z. Yu, "Multi-agent deep reinforcement learning for task offloading in group distributed manufacturing systems," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105710, Feb. 2023, doi:10.1016/j.engappai.2022.105710.

[62] A. B. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020. doi:10.1016/j.inffus.2019.12.012

[63] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti, "A Survey of Methods for Explaining Black Box Models," *ACM Computing Surveys*, vol. 51, no. 5, Article 93, 2018. doi:10.1145/3236009

[64] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.doi:10.1609/aimag.v40i2.2850

[65] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 70, pp. 3319–3328, 2017, doi:10.48550/arXiv.1703.01365

[66] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 70, pp. 3319–3328, 2017, doi:10.48550/arXiv.1704.02685

[67] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Lo-

calization," *International Journal of Computer Vision*, vol. 128, pp. 336–359, 2020. doi:10.1007/s11263-019-01228-7

[68] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, pp. 9525–9536, Dec. 2018, doi:10.48550/arXiv.1810.03292

[69] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, pp. 4766–4777, Dec. 2017, doi:10.48550/arXiv.1705.07874

[70] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016. doi:10.1145/2939672.2939778

[71] T. T. Nguyen, T. T. Huynh, Z. Ren, T. T. Nguyen, P. L. Nguyen, H. Yin, and Q. V. H. Nguyen, "Privacy-preserving explainable AI: a survey," *Science China Information Sciences*, vol. 68, no. 1, art. 111101, Jan. 2025, doi:10.1007/s11432-024-4123-4.

[72] P. Dubey and M. Kumar, "Integrating explainable AI with federated learning for next-generation IoT: A comprehensive review and prospective insights," *Computer Science Review*, vol. 56, art. 100697, 2025, doi:10.1016/j.cosrev.2024.100697

[73] Hong, D., Gu, Q., Whitehouse, K.: High-Dimensional Time Series Clustering via Cross-Predictability. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR, vol. 54, pp. 642–651 (2017)

[74] C. B. Do, "The multivariate Gaussian distribution," *CS 229 Lecture Notes*, Stanford University, Stanford, CA, USA, Oct. 2008.

[75] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018. ISBN 978-0262039246

[76] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014. ISBN 978-1118625873, doi:10.1002/9780470316887

[77] Bellman, R.: A Markovian Decision Process. *Journal of Mathematics and Mechanics* **6**(5), 679–684 (1957) doi:10.1512/IUMJ.1957.6.56038

[78] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992. doi:10.1007/BF00992698

[79] V. Mnih *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. doi:10.1038/nature14236

[80] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, 2001. ISBN 978-0471873396.

[81] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. doi:10.1109/COMST.2017.2745201

[82] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013, doi:10.1109/TWC.2013.072513.121842

[83] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," in *Handbook of Reinforcement Learning and Control*, Springer, 2021, pp. 321–384, doi:10.1007/978-3-030-60990-0_12

[84] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020. doi:10.1109/TCYB.2020.2977374

[85] M. L. Littman, "Markov Games as a Framework for Multi-Agent Reinforcement Learning," in *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pp. 157–163, 1994. doi:10.1016/b978-1-55860-335-6.50027-1

[86] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2–4, 2016.

[87] W. Cheng, X. Liu, X. Wang, and G. Nie, "Task offloading and resource allocation for industrial Internet of Things: A double-dueling deep Q-network approach," *IEEE Access*, vol. 10, pp. 103111–103120, 2022, doi:10.1109/ACCESS.2022.3210248

[88] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, New York, USA, Jun. 20–22, 2016, vol. 48, pp. 1995–2003.

[89] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," in *Proc. AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100. doi:10.1609/aaai.v30i1.10295

[90] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, pp. 1273–1282, 2017.

[91] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," arXiv preprint arXiv:1312.6034, Dec. 2013.

[92] SimPy Development Team, *SimPy: Discrete-Event Simulation for Python*, Available: `https://simpy.readthedocs.io/`, Accessed: Feb. 2026.

[93] Hong, D., Gu, Q., Whitehouse, K.: High-dimensional time series clustering via cross-predictability. In: Artificial Intelligence and Statistics, pp. 642–651. PMLR (2017)

[94] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, and H. Chen, "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis," in *Proc. of the ACM SIGCOMM Conference*, 2015, pp. 139–152. doi:10.1145/2785956.2787496

[95] M. G. Kapteyn, D. J. Knezevic, and K. E. Willcox, "Toward predictive digital twins via component-based reduced-order models and interpretable machine learning," in *Proceedings of the AIAA SciTech 2020 Forum*, Orlando, FL, USA, Jan. 2020, pp. 1–19, AIAA, doi:10.2514/6.2020-0418

[96] E. Çetin, C. Barrado, E. Salamí, and E. Pastor, "Analyzing Deep Reinforcement Learning Model Decisions with Shapley Additive Explanations for Counter Drone Operations," *Applied Intelligence*, vol. 54, pp. 12095–12111, 2024. doi:10.1007/s10489-024-05733-2

[97] Y. Wang, Y. Pu, and E. Wong, "A SHAP-based touch event prediction (SHAP-TEP) framework for explainable human-to-machine networks," in *Proc. IEEE Opto-Electron. Commun. Conf. (OECC)*, Melbourne, Australia, Jun./Jul. 2024, doi:10.1109/OECC54135.2024.10975446.

[98] A. H. Oveis, E. Giusti, G. Meucci, S. Ghio, and M. Martorella, "Explainability in hyperspectral image classification: A study of XAI through the SHAP algorithm," in *Proc. 13th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens. (WHISPERS)*, Athens, Greece, Oct./Nov. 2023, doi:10.1109/WHISPERS61460.2023.10430776.

[99] T. Hickling, A. Zenati, N. Aouf, and P. Spencer, "Explainability in Deep Reinforcement Learning: A Review into Current Methods and Applications," *ACM Computing Surveys*, vol. 56, no. 5, article 125, pp. 1–35, 2024. doi:10.1145/3623377

[100] C. Ferreira, K. N. Otto, W. Li, and B. Eisenbart, "Comparative study of LIME, SHAP and SHAP interaction quantification for improved FMEA development in energy-based maintenance," in *Proc. IEEE Int.*

Conf. Ind. Eng. Eng. Manag. (IEEM), Melbourne, Australia, Dec. 2025, doi:10.1109/IEEM63636.2025.11357827.