ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING

Ciclo 36

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Middleware-Enabled Frugality for Intelligent and Distributed Edge Applications

Presentata da:

Matteo Mendula

Coordinatore Dottorato
Prof.ssa Ilaria Bartolini

Supervisore
Prof. Paolo Bellavista

Esame Finale Anno 2024

to the beauty of small things,
capable of bringing such big changes

Abstract

In the ever-evolving landscape of computing, the integration of Artificial Intelligence has become pervasive, empowering applications with unprecedented capabilities. However, as the scope of AI expands, so do the challenges associated with resource constraints, particularly in the realm of intelligent and distributed Edge Applications. This thesis delves into the intersection of two concepts—Frugality and Serverless paradigms—aiming to revolutionize the deployment and functionality of intelligent agents on constrained devices.

Distributed Computing in all its facets is discussed, addressing the main challenges related to device heterogeneity and seamless networking management. Experimental contributions in edge industrial scenarios are presented [26], confirming the practical application of digital twin technology. In addition, a smart routing approach is novelly presented [27] showcasing the beneficial application of AI on Distributed Computing tasks.

The exploration extends to intelligent and distributed Edge Applications, where the fusion of AI and decentralized computing offers unlimited potential for swift responsiveness and restrained energy consumption. Understanding the importance of power efficiency in constrained environments, this thesis places a particular emphasis on the trade-off between performance and power consumption. Greener and more sustainable approaches to Distributed Learning are proposed [10, 180], taking into account power consumption in Federated Learning round planning strategies.

Then, we address the challenges related to data constrained scenarios. Privacy and Trust related issues are addressed in a Federated Learning setting proposing a novel contribution [175]. Additionally, considering the scarcity of data and model complexity, we introduce an ensembling of weak autoregressor as a striking solution for traffic volume forecasting [181].

During the exploration at the intersection of AI, frugality, and Serverless Computing, we then focus our attention on those cases where networking conditions are unstable and unreliable. Following the Split Computing paradigm we present a distilled encoder [183] capable of challenging the performance of deeper neural networks, enabling realtime semantic compression on mobile devices.

Table of Contents

List of Figures

List of Tables

List of Publications

[10] A. Agiollo, M. Mendula, P. Bellavista, A. Omicini.. (2024). EneA-FL: Energy-aware Orchestration for Serverless Federated Learning. *Future Generation Computer Systems*.

[183] M. Mendula, S. L.G. Contreras, M. Levorato, P. Bellavista.. (2024). Furcifer: a Context Adaptive Middleware for Real-world Object Detection Exploiting Local, Edge, and Split Computing in the Cloud Continuum". *PerCom, Biarritz, France, 2024*.

[175] C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, P. Bellavista.. (2023). TruFLaaS: Trustworthy Federated Learning as a Service. *IEEE Internet of Things Journal*.

[181] M. Mendula, A. Bujari, L. Foschini, P. Bellavista.. (2022). A Data-Driven Digital Twin for Urban Activity Monitoring. *2022 IEEE Symposium on Computers and Communications (ISCC)*.

[180] M. Mendula, P. Bellavista.. (2022). Energy-aware Edge Federated Learning for Enhanced Reliability and Sustainability. *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*.

[27] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula and M. Picone.. (2022). Digital twin oriented architecture for secure and QoS aware intelligent communications in industrial environments. *Pervasive and Mobile Computing*.

[26] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula and M. Picone.. (2021). Application-Driven Network-Aware Digital Twin Management in Industrial Edge Environments. *IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7791-7801, Nov. 2021*.

[182] M. Mendula, S. Khodadadeh, S.S. Bacanli, S. Zehtabian, H.S. Sheikh, L. Bölöni, D. Turgut, P. Bellavista.. (2019). Interaction and Behaviour Evaluation for Smart Homes: Data Collection and Analytics in the ScaledHome Project. *MSWiM '20, Alicante, Spain*.

Chapter 1

Frugality Middleware Technologies for Edge and Fog Computing

Resource constrained scenarios represent one of the main challenges of nowadays Software Engineering. High level Design Patterns [91] for Object Oriented architectures give systems reusability and extensibility characterists meant for handling outstanding computing capabilities in sub-optimal networking conditions and with abundant energy budgets. In those cases, Service Oriented Architecture (SOA) [259] emerges as the well-established cornerstone. Given the ample availability of resources, the primary focus revolves around data access control, transparent usage, and maintainability. These aspects rightfully take center stage in the pursuit of effective solutions for systems operating in resource-rich environments.

The integration of Cloud Computing [267] and DevOps [76] practices further elevates the transparency and modularity of existing solutions. On one side Cloud Computing offers a flexible and on-demand resource pool, enabling diverse applications to seamlessly adapt to fluctuating workloads. On the flip side, Continuous Delivery (CD) and Continuous Integration (CI) offered by DevOps ensure the perpetual accessibility of remote services, simplifying the day-to-day tasks of insiders. Thanks to those advancements, everyone from all over the world is now capable of consuming fresh multimedia content with high Quality of Service (QoS) requirements.

All these advancements raise a paradigm-shifting question: where does this data come from? Considering that the greatest part of downstream internet traffic is composed of images and videos [11, 84] it is reasonable to infer that data-centers do not serve as the primary data sources. Usually both Human to Machine (H2M) and Machine to Machine (M2M) take place at the edges of the network where the level of computational capabilities, network quality and energy budgets decrease dramatically. Here high level programming languages are replaced by lower ones and the applicability of transparent frameworks is limited by a careful thriftiness in terms of floating-point operations per second (FLOPS), bandwidth and power usage. As the total number of connected sensors and devices grows [12, 65] the current way employed to collect and process data will be ineffective in handling the incoming amount of information transferred over the network. New requirements in terms of energy awareness must be addressed in a more distributed and flexible way, taking into account Edge [237] and Fog [289] technologies to tackle Big Data [268] related concerns. Stringent demands for low latency and high-quality services necessitate a computational shift toward the network's peripheries, aligning with the locations of end-users and data sources.

In this context, Big Data and Artificial Intelligence [114] can be viewed as two complementary facets of a coin. Progressing with one while neglecting the other is impractical, as Big Data stands as one of the most valuable assets of our times and Artificial Intelligence serves as the means to unlock its immense potential. The symbiotic relationship between these two domains is essential for harnessing the full capabilities of data-driven insights and advancing technological frontiers. As is often the case in engineering endeavors, superior results typically incur higher costs. In the context of AI applications, its promising high-level outcomes bring an added strain on existing infrastructures, which are tasked with ensuring the continual availability of both local and remote services across the network. In these scenario, moving as much computation as possible to the Edge appears to be the only viable solution.

These two trends–increased transparency facilitated by Cloud Computing and reduced latency provided by Edge Computing–collide in the realm of Fog Computing scenarios. a diverse array of heterogeneous devices, each possessing varying communication and computation capabilities, constitute the backbone of the infrastructure. While the modularity inherent in Cloud architectures might be desirable on one front, its universal applicability is not entirely feasible in this context. The reason lies in the necessity for more optimized strategies when managing limited resources within the Fog Computing paradigm. Therefore, it is imperative to emphasize the application of AI technologies at the edges of the network. This must be executed with a keen awareness of resource constraints, ensuring that the computational expenses associated with substantial additional processing remain constrained. Furthermore, there is the need to encapsulate the intricacies of optimization strategies, aligning them with the Serverless paradigm [47]. This cohesive approach aims to establish a Cloud Continuum that seamlessly bridges the gap between more powerful and resource-constrained devices.

This thesis embarks on a comprehensive exploration of the evolving landscape of computing, with a central focus on the integration of Artificial Intelligence (AI) and its impact on resource-constrained environments, particularly in the domain of intelligent and distributed Edge Applications. The first part of the thesis delves into the core concepts of Transparent Frugality and the fusion of AI with Edge and Fog Computing, respectively. The former advocates for optimal resource efficiency in AI functionalities, while the latter explores novel approaches to enhance frugality in the context of edge devices. By delving into these dimensions, the research aims to uncover innovative approaches that not only enhance frugality but also navigate the intricate landscape of performance-energy tradeoffs in a way that does not compromise the expected intelligence and responsiveness of modern applications. Then, we introduce a groundbreaking proposition of an energy-aware serverless paradigm tailored for intelligent agents on constrained devices, redefining the conventional boundaries of serverless computing to address unique challenges. In this context, Federated Learning [25, 154, 285] is used as a practical use case to explore the multi-dimensional trade-off between resource optimization and QoS performance. Lastly, both theoretical insights and practical solutions, with the ultimate goal of shaping the future of intelligent computing in resource-constrained environments. The overarching theme throughout the thesis is the pursuit of a more efficient, transparent, and frugal integration of AI, laying the foundation for intelligent applications at the edge of our computing landscape.

1.1 Structure of the Thesis

Following a concise introduction to the motivations driving this thesis, this chapter succinctly outlines the unique contributions made by the candidate that surpass the current state of the art. The subsequent sections of this document draw upon content from papers that have been either published or are currently under review.

Chapter 2 provides background information about Distributed Computing and computation offloading, with a literature review of the principal challenges about microservice orchestration in resource constrained setting. The applicability of the digital twin (DT) approach within industrial IoT environments is considered, presenting a novel middleware for dynamically and seamlessly managing network resources in edge industrial environments.

Chapter 3 focuses on the symbiotic interplay between AI-enhanced applications and Edge environments. The main definitions about Frugality and energy aware AI are provided with references to the current state of the art. Both AI for Edge and AI on Edge [68] are taken into account, proposing experimental contributions for each scenario.

Chapter 4 focuses on another pivotal aspect of Frugaly: AI with limited data. In the broad context of AI-applications Big Data [269] availability persists as a mandatory requirement across the majority of use cases. Unfortunately rich and meaningful dataset are not always accessible, and even when present they comes with a very high price. This section explores Small Data approaches as an alternative to canonical ones, tackling privacy related issues when limited data are distributed across participants and how low complexity algorithms can in those cases outperform deeper neural networks. The section features two experimental contributions: one centered on a blockchain-based privacy framework and the other applying low-complexity algorithms in the context of smart city traffic prediction.

Chapter 5 explores the beneficial consequences of Semantic Compression and Split Computing as a Frugality-oriented technique for power and bandwidth saving. A middleware for containerized and ML-based computer vision applications is originally introduce as a real world use case testing scenario. Model quantization [99] and Knowledge Distillation [117] are employed here to further optimize resource usage on edge devices with limited computing capabilities.

Chapter 6 serves as the conclusion of this thesis, providing a comprehensive summary of the research findings. It not only encapsulates the key insights derived from the present work but also elucidates the most promising avenues for future research, both from a broader perspective and specifically stemming from the author's undertaken efforts.

1.2 Challenges

In the ever-evolving landscape of computing, two distinct challenges emerge as pivotal considerations for system architects and developers alike. The first challenge centers on the perpetual trade-off between achieving higher performance and managing resource consumption efficiently. This delicate balancing act poses a conundrum: as we strive to enhance system performance, the corresponding increase in resource utilization, including power consumption and potential scalability constraints, becomes an inevitable consequence. Navigating this trade-off becomes crucial, requiring thoughtful optimization strategies that align performance goals with the available resources, striking a harmonious equilibrium between efficiency and capability. This is particularly true when limited capabilities in terms of computation, communication and available energy come into play. Efficient algorithms, lightweight processing, and intelligent power management strategies become indispensable tools in addressing this challenge.

The second challenge delves into the intricacies of optimization and transparency within the targeted software architectures. Here, the focus lies on streamlining software layers by minimizing unnecessary components and eschewing the temptation of introducing excessive "syntactic sugar". This approach aims to cultivate a more transparent and comprehensible software structure while fostering optimization. In practice, it involves the judicious use of high-level APIs to leverage backbone services, emphasizing simplicity and clarity in design without sacrificing performance. The goal is to create systems that are not only efficient and scalable but also transparent and maintainable, thereby addressing the multifaceted demands of contemporary computing environments. By successfully navigating these dual challenges, developers can craft systems that deliver optimal performance, resource utilization, and maintainability in the face of the ever-increasing complexities inherent in modern computing paradigms.

1.3 Contributions beyond the state of the art

Conceptual Contributions:

- In [27] and in Chapter 2 we explore the major contributions available in Fog and Cloud applications for Industrial IoT [38], delving into finer details the concept of Digital Twin (DT) [24] as an innovative technology expected to transform the industrial and manufacturing ecosystems. In addition, the recent state of the art about Software-defined networking (SDN) [277] is originally presented. Here AI applicability for traffic management is considered for anomaly detection and packet routing policies.

- Novelly, in Chapter 3 we analyse the state-of-the-art the Frugal [104] and Green [230] AI directions under a two-dimensional agenda: analyzing separately the interplay of AI for Edge and AI on Edge. On one side we delve into the application of AI to distributed computing problems. While on the other side, the trade-off between minimal resource utilization and optimal performance spans across the current literature highlighting the most remarkable and novel contributions.

Experimental Contributions:

- In [26] we propose a novel Application-driven Digital Twin Networking (ADTN) middleware to support the twofold objective of simplifying the interaction with heterogeneous distributed industrial devices and of dynamically managing network resources. Here we discuss the support for adopting the Digital Twin (DT) abstraction, emphasizing its role in simplifying interactions with physical devices through Simple Digital Twins (SDTs). These SDTs serve as protocol gateways, facilitating communication with devices via various protocols. Additionally, the concept of Composed Digital Twins (CDT) is introduced, highlighting the representation of complex industrial applications as dynamic coordination among multiple SDTs. The deployment of SDTs on edge nodes is proposed, with each node managing multiple SDTs related to neighboring devices, creating a multi-hop multi-path topology. This approach aims to simplify network resource management by utilizing a high-level representation of industrial applications to dynamically and autonomously manage resources within production sites. Ultimately, this solution proved on a real testbed to enhance the efficiency and safety of industrial networks by enabling faster configuration for different applications within a production site or across multiple sites. Chapter 2 is partially based on this work.

- In [27] we highlight the need for enhanced safety and security measures in modern networking industrial environments, emphasizing the importance of IEC 62443 zones and conduits [151] for clear separation and communication security. To address challenges in implementation, the Digital Twin approach is proposed for zones and conduits, simplifying the definition and management of inter-machine security requirements. The use of an intelligent reasoner is recommended for real-time monitoring and reconfiguration, allowing for a dynamic trade-off between security and performance, with demonstrated feasibility and efficiency in industrial environments.

- In [180] we introduce a groundbreaking distributed framework designed to collect real-time and detailed process metrics for intelligent process management. Our focus lies in exploring pertinent FL round planning strategies, which are categorized as either static, involving a predetermined number of iterations, or dynamic, where the iteration count is decided at runtime based on trade-offs between model accuracy and energy expenditure. To substantiate the efficacy of our approach, we outline the design of an experimental testbed. This testbed demonstrates the implementation of an Energy-Aware Federated Learning (EFL) process using authentic data, highlighting the capabilities of our solution in terms of metric collection and adaptive planning based on the observed data. Chapter 3 focuses on the main challenges in that direction presenting the experimental contribution we developed in relationship with the current state of the art.

- In [10] we introduce EneA-FL an innovative scheme, termed EneA-FL, for serverless smart energy management. This novel approach dynamically adapts to optimize the training process, promoting seamless interaction between Internet of Things (IoT) devices and edge nodes. The proposed middleware incorporates a containerized software module that efficiently manages the interaction of each worker node with the central aggregator. EneA-FL makes informed decisions about node inclusion in subsequent training rounds by monitoring local energy budgets, computational capabilities, and target accuracy. This intelligent approach effectively balances the tripartite trade-off between energy consumption, training time, and final accuracy. Through a series of extensive experiments across diverse scenarios, our solution showcases impressive results, achieving a notable reduction of between 30% and 60% in energy consumption compared to popular client selection approaches in the literature. Furthermore, it proves to be up to 3.5 times more efficient than standard Federated Learning solutions.

- In [175] a groundbreaking blockchain-based architecture to establish Trustworthy Federated Learning as a Service (FLaaS). Our solution integrates blockchain, smart contracts, and a Decentralized Oracle Network (DON) to create a collaborative and secure AI model training system resilient to attacks from both the server and malicious participants. To ensure the validation process's integrity, we design a novel protocol relying on smart contracts and the DON. The DON dynamically supplies the smart contract with a subset of the validation dataset. Furthermore, we put forth a weighted aggregation strategy that considers each participant's trust level. Taking into account their historical performance in preceding rounds, each client is assigned a trust level, enhancing the fairness and reliability of contributions in the federated learning framework.

- In [183] we introduce and extensively evaluate Furcifer, an innovative middleware framework for Computer Vision tasks. Tailored to seamlessly adapt to the computing demands of highly dynamic environments, Furcifer transparently monitors the underlying system's state and dynamically predicts the feasibility of (Edge Computing) *EC*, (Local Computing) *LC*, and (Split Computing) *SC* [173] configurations. The core of Furcifer adopts a novel containerized approach, enabling low-overhead transitions between computing modalities and data processing modules. Notably, Furcifer achieves a context switch latency of less than 2ms and utilizes minimal storage – approximately 7GB upon instantiation with less than 0.3% transmitted over the network during runtime. The system monitoring module introduces minimal overhead and embeds algorithms for lightweight and efficient system state analysis. Our evaluation, focusing on Object Detection (OD) as a use case, involves over 250 experiments with various wireless technologies, showcasing Furcifer's ability to dynamically adapt configurations. Results demonstrate a remarkable 2x reduction in energy consumption, a 30% increase in mean Average Precision compared to static *LC*, and a three-fold rise in frame per second rate compared to static *EC* full offloading. Furthermore, Furcifer's highly optimized *SC* module surpasses state-of-the-art practical *EC* and *SC* configurations in certain parameter regions, emphasizing the significance of *SC* in the array of available computing configurations.

Chapter 2

Computing Paradigms Close to the Edge, Challenges and Possible Solutions

In the contemporary landscape of information technology, data stands as the primary commodity, and the possession of larger datasets often translates into increased value for data-centric businesses. As per the International Data Corporation (IDC), the volume of digital data surpassed 1 zettabyte in 2010 [92], and since 2012, a staggering 2.5 exabytes of new data are generated daily [176].

With the escalating velocity and volume of data, the conventional approach of transferring large datasets from IoT devices to the cloud faces efficiency challenges and, in certain instances, becomes impractical due to bandwidth limitations. Simultaneously, the emergence of time-sensitive and location-aware applications, such as patient monitoring, real-time manufacturing, self-driving cars, drone swarms, and cognitive assistance, presents a scenario where the distant cloud infrastructure struggles to meet the ultra-low latency requirements and provide location-aware services. Additionally, the scalability of cloud solutions may be inadequate to handle the vast amount of data generated by these applications. Furthermore, privacy concerns in specific applications make sending data to the cloud an unviable solution, necessitating alternative strategies for data processing and storage closer to the source.

To tackle the challenges posed by high-bandwidth, geographically-dispersed, ultra-low latency, and privacy-sensitive applications, there is an imperative need for a computing paradigm situated closer to connected devices. Both industry [43] and academia [36] have proposed Fog computing to address these issues and fulfill the demand for a computing paradigm in proximity to connected devices. Fog computing serves as a bridge between the cloud and IoT devices, facilitating computing, storage, networking, and data management on network nodes situated close to IoT devices. This implies that computation, storage, networking, decision-making, and data management occur along the path between IoT devices and the cloud as data moves from IoT devices to the cloud. The research community has also introduced other similar computing paradigms such as mist computing [72, 89], cloud of things [5], and cloudlets [18, 198], all aimed at addressing the aforementioned challenges in various ways.

In Section 2.1 we delve into a comparative analysis of Fog computing and its counterparts, illustrating the inherent advantages of Fog computing across diverse use cases. The primary obstacles hindering the widespread adoption of Edge computing are meticulously enumerated and scrutinized. These challenges encompass issues of trust and authentication, the imperative need for eco-friendly Fog computing solutions, scalability concerns, considerations regarding bandwidth optimization, and promising direction towards intelligent and autonomous policy management on the Edge. Moreover, a groundbreaking approach, termed Frugality, is introduced as a disruptive solution. Section 2.2 focuses on the adoption of the Digital Twin paradigm as a way to apply frugality in heterogeneous IIoT environments. Then, in Section 2.3 we present our novel contribution for intelligent and autonomous packet routing.

2.1 From the Cloud to the Far Edge

In the ever-evolving landscape of computing paradigms, the journey from centralized cloud environments to the far edge represents a transformative shift that aligns with the evolving demands of contemporary applications. This section explores the paradigm shift from cloud to the far edge, delving into emerging computing models that bring processing capabilities closer to connected devices. We will navigate through Fog Computing, edge computing, and other related paradigms, examining their distinctive features, advantages, and implications for the future of computing architecture. As we traverse this landscape, we uncover how these advancements are reshaping the way applications are deployed, managed, and optimized in the era of pervasive connectivity and data-driven innovations.

2.1.1 Traditional Cloud Computing

Cloud computing has played a pivotal role in extending the accessibility and capabilities of computing, storage, and networking infrastructure to various applications. The National Institute of Standards and Technology (NIST) defines Cloud Computing as a model that facilitates ubiquitous, on-demand network access to shared computing resources [179]. Cloud data centers, characterized by large pools of highly accessible virtualized resources, offer dynamic reconfigurability to accommodate scalable workloads. This adaptability is particularly advantageous for cloud services, which often adopt a pay-as-you-go cost model [262]. The pay-as-you-go cost model enables users to conveniently access remote computing resources and data management services, paying only for the resources they consume. Major cloud providers, including Google, IBM, Microsoft, and Amazon, play a crucial role by furnishing and provisioning expansive data centers to host these cloud-based resources. This approach not only enhances flexibility but also aligns with a more cost-effective and scalable utilization of computing infrastructure.

Cloud computing encompasses four primary deployment models: private cloud, community cloud, public cloud, and hybrid cloud [179]. Private clouds are tailored for singular entities, ensuring high privacy and configurability, resembling traditional company-owned server farms but lacking the pay-as-you-go cost model. Community clouds serve a collective user base, with decentralized ownership shared among multiple organizations within the community. Public clouds, offered by providers like Amazon, IBM, Google, and Microsoft, are widely popular, easy to maintain, and cost-effective, although they may lack full customization. Hybrid clouds combine various deployment types, offering finer control over virtualized infrastructure through standardized or proprietary technology [247].

The cloud offers a spectrum of services, categorized as infrastructure, platform, and software (IaaS, PaaS, SaaS) - see Table 2.1 as a reference - catering to the diverse needs of application developers. In the realm of Infrastructure as a Service (IaaS), cloud consumers gain direct access to IT infrastructures encompassing processing, storage, and networking resources [70] inside a virtualized and isolated environment. In the platform as a service (PaaS) model, cloud providers furnish a scalable and managed environment encompassing not only the underlying infrastructure but also tools and services for developers to build, test, and deploy applications efficiently. PaaS abstracts complexities associated with infrastructure management, allowing developers to focus primarily on application code and functionality. This model accelerates the development lifecycle by offering ready-made services such as databases, development frameworks, and runtime environments, enabling developers to create and deploy applications without being burdened by the intricacies of infrastructure configuration. Software as a Service (SaaS) is a Cloud Computing service model where users access and utilize fully functional software applications over the internet, eliminating the need for local installations and maintenance. In the SaaS model, the cloud provider manages all aspects of the software, including infrastructure, maintenance, updates, and security. SaaS is particularly advantageous for businesses, offering cost-effective solutions, streamlined workflows, and efficient access to a wide range of applications without the complexities of traditional software management.

In terms of infrastructure control, IaaS provides the highest level of control, enabling users to manage virtualized infrastructure components such as servers, storage, and networking. While this grants flexibility, it also entails greater responsibility for users in terms of configuration and maintenance. Any additional layer increases the overhead over the infrastructure and may result in more complex management tasks.

Resource provisioning and limitations

The demand for cloud resources is another relevant aspect. This can be highly dynamic, leading to challenges with fixed resource allocation that can result in either over-provisioning or under-provisioning, as illustrated in Fig. 2.1. Cloud Computing's fundamental principle revolves around provisioning precisely the necessary resources based on demand. This involves leveraging virtualization for on-demand application deployment and employing resource provisioning to effectively manage hardware and software in cloud data centers. The topic of resource provisioning is extensively explored in Cloud Computing [42, 48, 122, 243, 292], given its significance in optimizing performance and cost. Due to the inherent difficulty in accurately predicting service usage from tenants, most cloud providers adopt a pay-as-you-go payment model. This approach allows providers to be flexible in resource provisioning, ensuring that clients only pay for the actual amount of resources they consume, aligning with the core tenets of efficiency and cost-effectiveness in Cloud Computing. The concept of cloud

Table 2.1: In summary, IaaS focuses on managing infrastructure components, PaaS abstracts even more of the underlying infrastructure to provide a platform for application development, and SaaS offers fully managed applications delivered over the internet. The classification is based on the level of abstraction and the extent of control and responsibility delegated to users or handled by the cloud provider within the application stack.

On-Premises	IaaS	PaaS	SaaS
🟥 Application	🟥 Application	🟥 Application	🟦 Application
🟥 Data	🟥 Data	🟥 Data	🟦 Data
🟥 Runtime	🟥 Runtime	🟦 Runtime	🟦 Runtime
🟥 Middleware	🟥 Middleware	🟦 Middleware	🟦 Middleware
🟥 OS	🟥 OS	🟦 OS	🟦 OS
🟥 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization
🟥 Server	🟦 Server	🟦 Server	🟦 Server
🟥 Storage	🟦 Storage	🟦 Storage	🟦 Storage
🟥 Networking	🟦 Networking	🟦 Networking	🟦 Networking

RED: managed by User

BLUE: managed by Vendor/Service provider

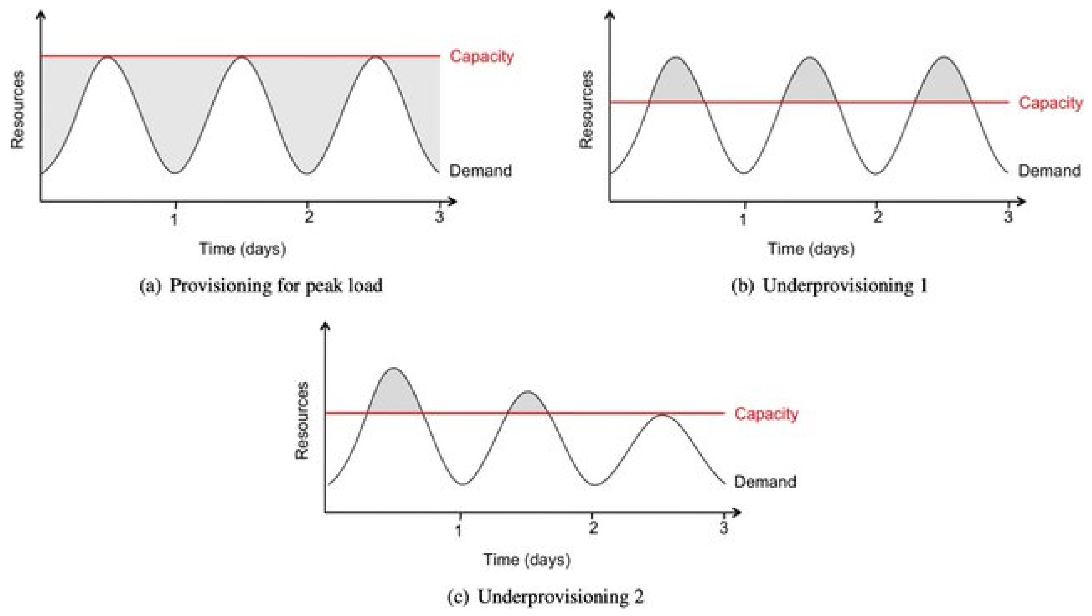


Figure 2.1: Shaded regions of (a) Over provisioning with resource underutilization; (b) Case 1 of under provisioning with compromised potential revenue from end-users; (c) Case 2 of under provisioning with attrition of cloud end-users.

provisioning involves three distinct cases [188], each depicting different scenarios related to resource allocation:

- Over provisioning with resource under utilization: This case illustrates a situation where cloud resources are allocated in excess of actual demand. Consequently, there is under utilization of resources, leading to inefficiency and increased operational costs.

- Under provisioning with compromised potential revenue from end-users: In this scenario, the cloud resources provided fall short of the actual demand, resulting in under provisioning. Part of the user demand is unmet, leading to compromised potential revenue from end-users. This situation emphasizes the importance of accurately aligning resource allocation with user requirements to avoid revenue loss.

- Under provisioning with attrition of cloud end-users: The shortfall in resources may lead to dissatisfaction among users, causing them to discontinue their engagement with the cloud service.

While Cloud Computing initially aimed at providing ubiquitous access to computing resources, the latency associated with accessing cloud-based applications might be impractical for certain mission-critical or low-latency-sensitive applications. The surge in data generation at the network edge necessitates cloud resources to be closer to where data is produced. The demand for high-bandwidth, geographically-dispersed, low-latency, and privacy-sensitive data processing has led to the emergence of computing paradigms closer to connected devices. Fog computing, proposed by both industry and academia, addresses these requirements by bringing computing resources closer to the edge. To facilitate a detailed comparison among various Fog Computing-related paradigms, we introduce different computing models, starting with Fog Computing.

2.1.2 Edge Computing: Demystifying an Umbrella Term

Edge computing extends the capabilities of connected devices, enhancing the management, storage, and processing power of generated data. Edge computing is strategically positioned at the edge of the network in close proximity to Internet of Things (IoT) devices. It is important to note that the edge is not directly on the IoT devices but is typically within one hop [238]. Although, in local IoT networks, the edge may be more than one hop away from IoT devices. OpenEdge Computing Whitepaper [60] defines edge computing as computation conducted at the network's edge through small data centers that are in close proximity to users. The primary vision for edge computing is to furnish compute and storage resources close to users in an open standards and ubiquitous manner. This paradigm plays a crucial role in the IoT landscape by intelligently filtering, preprocessing, and aggregating IoT data through cloud services deployed near IoT devices [212]. Edge computing excels in addressing issues such as privacy, latency, and connectivity. Due to its proximity to users, latency in edge computing is typically lower than Cloud Computing, provided sufficient local computation power is available. Service availability is also higher in edge computing as connected devices are not constrained by the limitations of traditional mobile computing resources or the wait times associated with highly centralized platforms. Edge computing incorporates small data centers, contributing to higher service availability. Furthermore, edge computing can expand its computing capabilities beyond MACC by adopting hybrid architectures that integrate peer-to-peer and cloud computing models. This flexibility positions edge computing as a versatile and powerful paradigm in the evolving landscape of connected devices and IoT.

It is important to note that terms like edge computing, cloudlets, fog computing, and mist computing (discussed in at the end of Section 2.1.5) are sometimes used interchangeably in literature, given their shared use of the term "Edge". In the telecommunications industry, the term "Edge" typically refers to 4G/5G base stations, RANs, and ISP access/edge networks. However, in the context of the Internet of Things (IoT) landscape [212], the term "Edge" pertains to the local network housing sensors and IoT devices. Essentially, the Edge represents the immediate first hop from IoT devices, such as WiFi access points or gateways, rather than the IoT nodes themselves. As addressed in Section 2.1.5, when computation occurs on the IoT devices directly, this paradigm is termed mist computing. Notably, General Electric [79] highlights the distinction between fog computing and edge computing. Fog computing primarily concerns interactions among edge devices like RANs, base stations, or edge routers, while Edge Computing focuses on the technology associated with connected things, such as WiFi access points.

2.1.3 Fog Computing: Between the Cloud and End-Devices

The term "Fog computing" was introduced in 2012 by researchers associated with Cisco Systems [36, 97]. However, the idea of processing application logic and data at the edge is not a recent development. The concept of Edge computation had already surfaced around the 2000s [278]. Additionally, a related concept known as cloudlets was put forth in 2009 [131]. Both Cloudlets and Fog computing represent advancements of a similar concept, emphasizing processing at the edge level. While Cloudlets find application in mobile networks, Fog computing is specifically tailored for connected devices such as IoT, aligning with the broader concept of the Internet of Things [102]. These technological developments signify an evolution in computing paradigms, with a shift towards decentralized processing at the edge to enhance efficiency and responsiveness in diverse application scenarios. Fog computing and edge computing, although sharing the common objective of moving computation and storage closer to the network edge and end-nodes, are distinct paradigms. According to the OpenFog Consortium [43], Fog Computing is hierarchical and offers a comprehensive range of computing, networking, storage, control, and acceleration capabilities across the entire spectrum from cloud to devices. In contrast, edge computing tends to be confined to computing at the edge, without the broader scope encompassed by fog computing. (Refer to Fig. 2.2) In addition, as better explanation about the differences between Fog and Cloud Computing [56] clarifies that "fog is inclusive of cloud, core, metro, edge, clients, and things." Fog computing aims to establish a seamless continuum of computing services from the cloud to the end devices, avoiding the treatment of network edges as isolated computing platforms. The vision for fog computing is that of a horizontal platform capable of supporting common fog computing functions across various industries and application domains, extending beyond traditional telecommunications services [44]. This distinction underscores the hierarchical and inclusive nature of fog computing in contrast to the more localized focus of edge computing.

Another definition of Fog computing, comes from the OpenFog Consortium [43]. Here it is defined as a system-level architecture that horizontally distributes computing, storage, control, and networking functions. This distribution occurs along a continuum from the cloud to

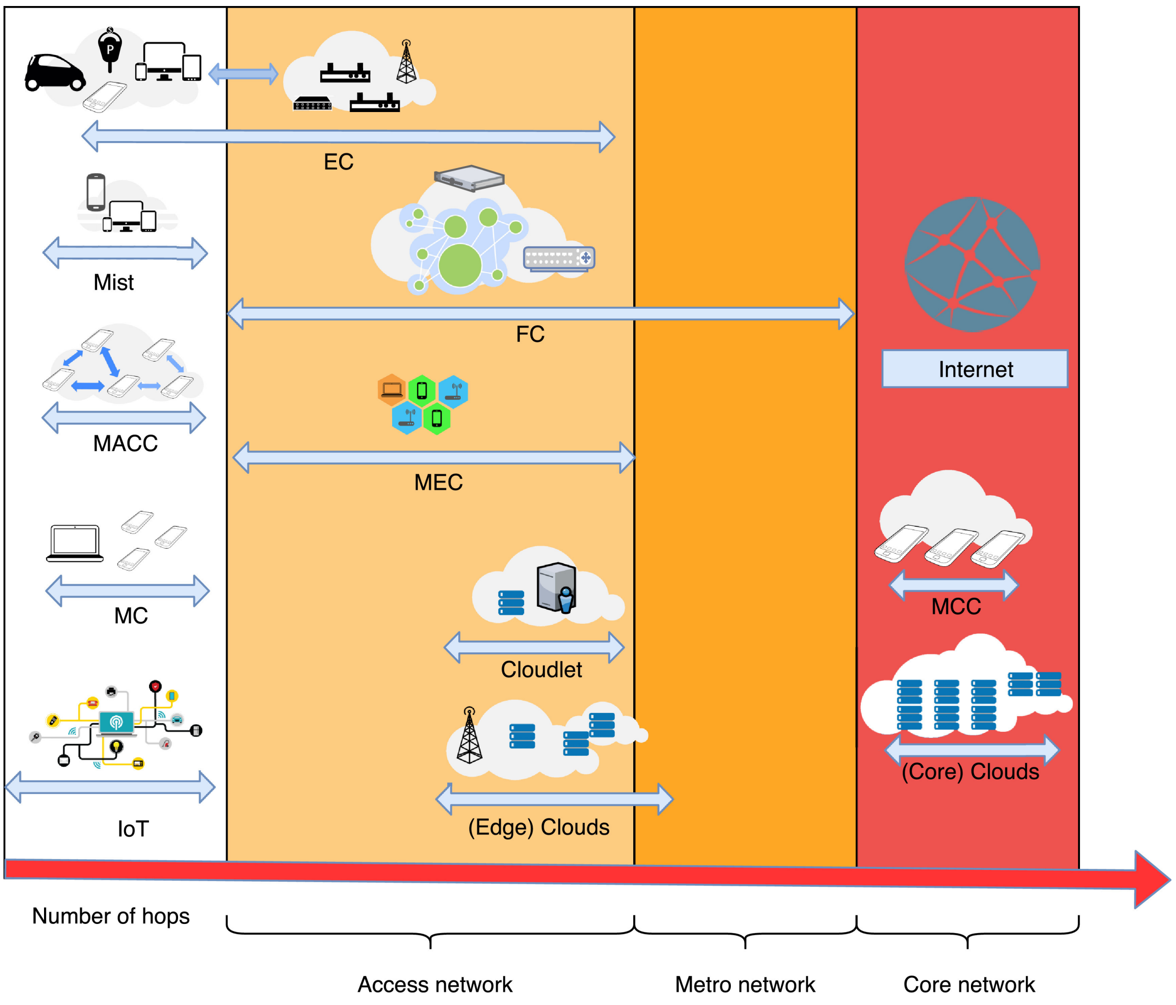


Figure 2.2: Analyzing fog computing and its associated computing paradigms concerning their positioning and proximity to the core cloud.

connected devices. Unlike vertical platforms that isolate applications in silos, Fog Computing's horizontal approach allows for the dispersion of computing functions across various platforms and industries. While a vertical platform may excel in supporting a specific application type, it lacks consideration for interactions between different vertically-focused platforms. Beyond its horizontal architecture, Fog Computing offers a flexible platform tailored to address the data-driven requirements of operators and users. Its primary objective is to provide robust support for the Internet of Things. Fog computing serves as a crucial link connecting the cloud and end devices, such as IoT nodes. It achieves this by facilitating computing, storage, networking, and data management on network nodes situated in close proximity to IoT devices. Consequently, the processes of computation, storage, networking, decision-making, and data management extend beyond the confines of the Cloud and unfold along the path from IoT devices to the cloud. This distributed approach ensures that these operations occur preferably in close proximity to the IoT devices themselves. As an example, in Intelligent Transportation Systems (ITS) [8], the compression of GPS data can take place at the edge, occurring before the transmission of data to the cloud.

Adavantages of Fog over Cloud

A commonly cited example to distinguish between fog and Cloud Computing revolves around the support for latency-sensitive applications while maintaining satisfactory quality of service (QoS). Fog nodes, strategically positioned near IoT source nodes, significantly reduce latency compared to traditional Cloud Computing. Although this example underscores the intuitive appeal of Fog Computing, it is important to note that latency-sensitive applications represent just one facet of the diverse range of applications that justify the adoption of Fog Computing. Unlike centralized cloud data centers, nodes in Fog Computing are dispersed in less centralized locations, offering a wide geographical presence in substantial numbers. In Fog Computing, security measures need to be implemented at the edge or in dedicated locations of fog nodes, in contrast to the centralized security mechanisms employed in dedicated buildings for cloud data centers. The decentralized nature of Fog Computing allows devices to either function as Fog Computing nodes themselves (e.g., a car acting as a fog node for onboard sensors) or utilize fog resources as clients of the fog. This decentralized deployment model enhances flexibility and responsiveness in catering to a variety of computing needs.

The primary distinctions between cloud and Fog Computing stem from the scale of hardware components associated with these computing paradigms. Cloud computing offers high availability of computing resources with relatively high power consumption, while Fog Computing provides moderate availability of computing resources at lower power expenditure [133]. Cloud computing typically relies on extensive data centers, whereas Fog Computing makes use of smaller-scale servers, routers, switches, gateways, set-top boxes, or access points. Due to the compact nature of Fog Computing hardware, it can be located in closer proximity to users. Fog computing plays a pivotal role in enabling devices to measure, monitor, process, analyze, and react by distributing computation, communication, storage, control, and decision-making closer to IoT devices [43]. This approach offers numerous advantages across various industries, including but not limited to energy management, manufacturing, transportation, healthcare, and smart cities. The application of Fog Computing in these sectors enhances efficiency, responsiveness, and real-time processing capabilities, contributing to the optimization and advancement of diverse industrial processes and services.

The distinctions and trade-offs between cloud and Fog Computing prompt the question of which one to choose. However, it is crucial to recognize that fog and Cloud Computing are not mutually exclusive; rather, they complement each other. The integration of cloud and Fog Computing allows for the optimization of services utilized by connected devices, creating a synergy that enhances overall performance. Federation between fog and cloud introduces advanced capabilities for data aggregation, processing, and storage. For instance, in a stream processing application, the fog can efficiently filter, preprocess, and aggregate traffic streams from source devices. Simultaneously, queries demanding heavy analytical processing or archival results can be directed to the cloud. An orchestrator, acting as a mediator, can facilitate the collaboration between cloud and fog. A fog orchestrator, for instance, can establish an interoperable resource pool, deploy and schedule resources for application workflows, and control quality of service (QoS) [272]. The incorporation of Software-Defined Networking (SDN) [145, 232, 276] empowers fog service providers with greater control over the network configuration, especially when managing a multitude of fog nodes responsible for data transfer between the Cloud and IoT devices. This collaborative approach leverages the strengths of both Fog and Cloud computing, optimizing the overall performance and responsiveness of connected systems.

Radio Access Networks (RAN) and Fog

Fog computing seamlessly integrates into mobile technologies through the concept of fog Radio Access Networks (F-RAN), providing a dynamic and efficient approach to network architecture. F-RAN leverages computing resources at the edge, enhancing performance in various ways, such as caching content for quicker retrieval and reducing the load on the front-haul. This integration is particularly relevant in the context of 5G-related mobile technologies [129]. F-RAN can be instrumental in optimizing content delivery and overall network efficiency. In contrast, Cloud Radio Access Network (C-RAN) takes a centralized control approach over F-RAN nodes. It utilizes virtualization to decouple base stations within a cell from their baseband functions, achieving greater flexibility and efficiency [52]. C-RAN involves deploying numerous low-cost Remote Radio Heads (RRHs) connected to a Base Band Unit (BBU) pool through front-haul links. Both F-RAN and C-RAN are well-suited for mobile networks with base stations and are considered as viable options for 5G deployments. Importantly, their implementation contributes to a more energy-efficient form of network operation, aligning with the broader goals of sustainability and resource optimization [201].

Figure 2.3 presents a classification of computing paradigms associated with Fog computing and illustrates the scope overlap among them. This figure serves as a visual representation of our comparison of Fog Computing and its related computing paradigms. To aid in understanding, Table 2.2 provides a list of acronyms used in this figure and throughout the paper. In

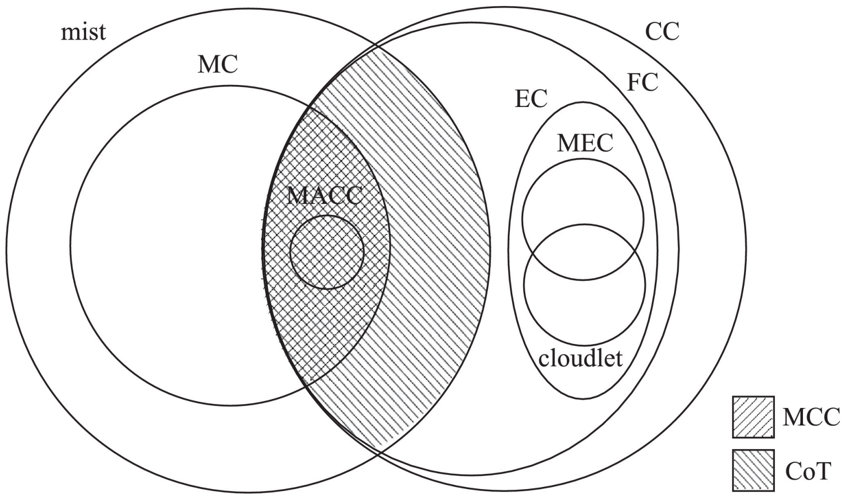


Figure 2.3: Fog-related Computing Paradigms. We define four unstructured knowledge integration mechanisms. A knowledge mapping mechanism is an instance of each knowledge integration mechanism. Lastly, knowledge validation defines three major requirements concerning the integration of unstructured knowledge.

the subsequent discussion, we explore these related computing paradigms in the order of their trend, highlighting how certain paradigms have paved the way for the emergence of others.

Table 2.2: Acronyms referring to Figure 2.3

IoT	Internet of Things	CC	Cloud Computing
MC	Mobile Computing	FC	Fog Computing
EC	Edge Computing	MEC	Multi-access Edge Computing
MCC	Mobile Cloud Computing	MACC	Mobile ad hoc Cloud Computing
CoT	Cloud of Things	mist	Mist Computing

2.1.4 Mobile Device Constraints at the Network Edge

The progress in Fog and Cloud Computing has been significantly influenced by the foundational work laid down during the development of Mobile Computing. Mobile computing [13], also known as nomadic computing, involves carrying out computing tasks through portable devices like laptops, tablets, mobile phones or embedded devices. This paradigm has paved the way for creating pervasive and context-aware applications, exemplified by features such as location-based reminders. At its core, Mobile Computing envisions adaptation within an environment characterized by low processing power and sporadic, limited network connectivity. The zenith of Mobile Computing technologies predates the emergence of Cloud Computing.

Numerous fundamental challenges inherent to Mobile Computing, including user mobility, network heterogeneity, and low bandwidth, were addressed in the literature before 2000. Solutions to these challenges have been found through advancements such as robust caching mechanisms, improvements in transmission hardware and protocols, and the development of efficient compression algorithms [85]. However, owing to the evolving demands of connected consumer devices and the increasing complexity of modern computing challenges, Mobile Computing alone may not be sufficient. This recognition has prompted the subsequent evolution and integration of fog and Cloud Computing paradigms to address the contemporary needs of a highly connected and data-intensive environment.

Fog and Cloud Computing have transformed the landscape of computing by liberating computation from the confines of local networks, effectively expanding the scale and scope of mobile computing. In traditional mobile computing, mobile devices can establish connectivity through technologies like Bluetooth [33], WiFi [167], ZigBee [158], and cellular protocols [14]. In contrast, Fog and Cloud Computing demand more resource-intensive hardware with virtualization capabilities. Security measures in mobile computing are typically implemented directly on the mobile devices themselves. In terms of available resources, if compared to fog and Cloud Computing, Mobile Computing is inherently more resource-constrained. However, recent years have witnessed substantial advancements in mobile hardware and wireless protocols, narrowing the resource gap considerably. These improvements have enhanced the capabilities of mobile computing, making it more robust and aligning it more closely with the evolving demands of contemporary computing environments. The strength main of Mobile Computing lies in its distributed computing architecture. Distributed applications benefit from this decentralized approach as mobile devices do not require a centralized location to operate. However, Mobile Computing comes with several drawbacks, including significant resource constraints, the delicate balance between autonomy and interdependence (prevalent in all distributed architectures), communication latency issues, and the necessity for mobile clients to efficiently adapt to changing environments [227].

Mobility:

The ability to change locations while connected to the network increases the volatility of some information. Certain data considered static for stationary computing becomes dynamic for mobile computing. For example, a stationary computer can be configured statically to prefer the nearest server, but a mobile computer needs a mechanism for determining which server to use. As volatility increases, cost-benefit trade-off points shift, calling for appropriate modifications in the design. For example, a highly volatile data object has fewer uses per modification.

For such objects it makes little sense to cache the data. As another example, consider static information, which is often managed by hand; to handle higher rates of change, automated methods are required. However, even where such methods exist, they may be ill-suited for the dynamism of mobile computing. Mobility introduces several problems: A mobile computer's network address changes dynamically, its current location affects configuration parameters as well as answers to user queries, and the communication path grows as it wanders away from a nearby server.

Power Consumption:

Batteries are the largest single source of weight in a portable computer. While reducing battery weight is important, too small a battery can undermine the value of portability by causing users to recharge frequently, carry spare batteries, or use their mobile computers less. Minimizing power consumption can improve portability by reducing battery weight and lengthening the life of a charge. Power consumption of dynamic components follows the following proportionality:

$$Power\_consumption \propto C \times V^2 \times F \tag{2.1}$$

Where:

- C is the capacitance of the circuit. This can be reduced by greater levels of VLSI integration [23] and multi-chip module technology [87];
- V is the voltage swing, which can be optimized by redesigning chips to operate at lower voltages;
- F is the clock frequency. This can be minimized by trading computational speed for power savings. To retain more computational power at lower frequencies, processors are being designed that perform more work on each clock cycle.

Power conservation in Mobile Computing is not only achieved through thoughtful design but also through efficient operational strategies. Power management software plays a crucial role in conserving power by selectively powering down individual components when they are idle. For example, this may involve spinning down the internal disk, turning off screen lighting or spin down the internal disk drive after it has been idle for just a few seconds [75]. Applications can also contribute to power conservation by adopting practices that reduce their consumption of computation, communication, and memory resources. This can be achieved by adjusting their operational behavior to perform periodic operations less frequently [184], thereby amortizing the start-up overhead and minimizing the overall energy consumption. Efficient power management at both the hardware and software levels is essential for optimizing the energy efficiency of mobile computing devices.

Low Bandwidth and Bandwidth Variability

Wireless networks inherently offer lower bandwidth compared to wired networks, necessitating careful consideration of bandwidth consumption in Mobile Computing designs. The deliverable bandwidth per user is influenced by the number of users sharing a cell, and the network's capacity can be evaluated based on its bandwidth per cubic meter. Two primary approaches can be employed to enhance this value:

- **Maintaining Multiple Cells at Different Frequencies:** This approach, while more flexible, is constrained by the available range of frequencies in the electromagnetic spectrum for public consumption;
- **Limiting Transmission Ranges:** By restricting transmission ranges, more cells can fit into a given area. This approach is preferred for its simplicity, reduced power requirements, and potential decrease in signal corruption. Transceivers covering smaller areas may achieve higher bandwidth.

Additionally, various techniques such as compression, logging (consolidating multiple short requests into larger ones), perfecting (anticipating files that will be needed soon), and write-back caching can help mitigate the impact of low bandwidth. Intelligent scheduling of communications further enhances system performance. Mobile Computing designs must contend with more significant variations in network bandwidth compared to traditional designs. An effective design should be adaptable to the currently available resources, offering users a variable level of quality. As a mobile element moves out of the range of one network transceiver, it seamlessly switches to another. Furthermore, there may be locations where users can access multiple transceivers operating on different frequencies.

Disconnections:

Wireless communication's susceptibility to disconnection poses a significant challenge in the design of successful Mobile Computing systems. Design strategies can focus on allocating resources to handle disconnections more elegantly or preventing them from occurring. In environments prone to frequent disconnections, it may be more effective for the mobile computer to function as a stand-alone unit rather than a mobile terminal. This involves splitting the application and user interface across the network. For wide-area networks, where round-trip Remote Procedure Call (RPC) [193] delays can be costly in terms of wasted processor clock cycles, operating asynchronously can make round-trip latencies and brief disconnections less expensive. Synchronous systems offer advantages such as a simpler design, implementation, and debugging. However, asynchronous models yield higher performance since the receiver does not block while waiting for requested data. Caching techniques [221] can enhance the performance of weakly-connected and disconnected operations. Still, preserving cache coherence under weak connectivity can be costly. The Coda file system addresses this challenge by maintaining cache coherence at multiple levels of granularity and utilizing callbacks. In the Coda solution, fast cache validation is achieved by comparing version stamps maintained by clients and servers, preserving validity through callbacks. This approach offers a trade-off between the precision of invalidation and the speed of validation.

Address Migration:

As individuals move, their Mobile Computers will utilize different network access points, or "addresses." However, current networking protocols are not inherently designed to handle dynamically changing addresses. Active network connections typically cannot be seamlessly moved to a new address. Once an address for a host name is known to a system, it is usually cached with a long expiration time, and there is often no mechanism to invalidate outdated entries. In the Internet Protocol, for instance, a host IP name is closely tied to its network address; changing locations implies acquiring a new IP name. Human intervention is commonly needed to coordinate address usage. To communicate with a Mobile Computer, messages must be sent to its most recent address. As Mobile Computers change locations, they utilize different network access points or "addresses." Several techniques can be employed to determine the current network address of a mobile unit [270]:

- **Selective Broadcast:** If a Mobile Computer is known to be in a set of cells, a message could be broadcasted to these known cells, asking the required mobile unit to reply with its current network address;
- **Central Services:** A logically centralized database holds the current addresses of all mobile units. Whenever a Mobile Computer changes its address, it sends a message to update the database;
- **Home Bases:** This is essentially a more centralized approach where only a single server knows the current location of a Mobile Computer;
- **Forwarding Pointers:** This method involves placing a copy of the new address at the old location. Each message is then forwarded along the chain of pointers leading to the Mobile Computer. This approach requires an active mechanism.

These drawbacks often render Mobile Computing unsuitable for current applications that demand low-latency or robustness, or that require large amounts of data to be generated, processed, and stored on devices. The challenges posed by resource constraints, adaptability to changing conditions, and communication latency can limit the applicability of Mobile Computing in scenarios where real-time responsiveness and extensive data processing capabilities are crucial.

2.1.5 Other Computing Paradigms

In addition to Mobile Computing, Cloud Computing, and Fog Computing, several other computing paradigms contribute to the diverse landscape of information technology. Each paradigm addresses specific challenges and offers unique advantages. In the following section we delve into finer details of possible alternatives to aformentioned computing strategies.

Mobile Cloud Computing

As Cloud Computing matured, it seamlessly integrated with Mobile Computing, giving rise to a valuable synergy known as Mobile Cloud Computing (MCC). In MCC, both data storage and data processing occur outside of the mobile device, expanding the reach of mobile computing applications beyond smartphones to a much broader spectrum of mobile subscribers [71]. The National Institute of Standards and Technology (NIST) extends this definition to encompass mobile devices, describing cloud computing as the synergy among IoT devices, mobile devices, and cloud computing that facilitates data-intensive and CPU-intensive applications in IoT environments [192]. Applications in MCC span various domains, including crowdsourcing [46], healthcare, sensor data processing (e.g., optical character recognition and image processing), and task offloading [215, 224]. A notable feature is the ability to dynamically partition mobile applications at runtime, enabling computationally intensive components to be adaptively offloaded to the cloud [34]. This dynamic offloading enhances the overall performance and capabilities of mobile applications, leveraging the resources and scalability provided by cloud computing infrastructure.

In the realm of Mobile Cloud Computing (MCC), the resources within mobile devices can harness the capabilities of resource-rich cloud services. This paradigm involves a substantial shift of computation from mobile devices to the cloud, offering several advantages. MCC is particularly adept at running computation-intensive applications, contributing to increased battery life for mobile devices. MCC brings together characteristics from both mobile computing and cloud computing, providing a unique blend of capabilities. By combining the objectives of these two paradigms, MCC ensures high availability of computing resources, a departure from the resource-constrained nature of traditional mobile computing. This expanded resource availability facilitates the emergence of high-computation applications, such as mobile augmented reality. The availability of cloud-based services in MCC is significantly higher than that of standalone mobile computing. Much like cloud computing and fog computing, MCC relies on cloud services to execute high-computation tasks. Notably, computation in MCC can be managed by both mobile devices and the cloud. Similar to cloud computing, security in MCC must be provisioned for both mobile devices and the cloud. This ensures a comprehensive security framework, addressing potential vulnerabilities in both local mobile environments and the broader cloud infrastructure. The synergistic integration of mobile computing and cloud computing in MCC contributes to enhanced computational capabilities, resource availability, and overall performance for mobile applications.

MCC, despite its advantages, inherits certain limitations from both mobile computing and cloud computing. Firstly, the centralized architecture in MCC, while efficient for sharing computation resources, may not align well with applications that prioritize the widespread presence of devices. The concentration of computation resources in a centralized manner might not be ideal for scenarios where device pervasiveness is a crucial consideration. Secondly, both cloud computing and MCC rely on cloud-based services, necessitating continuous Internet connectivity for access through WAN connections. This persistent requirement for an Internet connection poses a challenge, as applications running on these platforms demand uninterrupted access to the Internet. Unlike traditional mobile computing, where computation predominantly occurs on the device, MCC shifts the majority of computation to the cloud. This shift introduces connectivity challenges that were not prevalent in the context of standalone mobile computing, highlighting the dependence on a consistent and reliable network connection for optimal performance.

Mobile Ad Hoc Cloud Computing

Despite the widespread applicability of Mobile Cloud Computing (MCC), this paradigm may not always be suitable for scenarios lacking infrastructure or a centralized cloud. In such situations, an ad hoc mobile network presents itself as a viable alternative. Ad hoc mobile networks consist of nodes that dynamically form temporary networks through routing and transport protocols, representing the most decentralized form of a network [126]. In this context, mobile devices within an ad hoc mobile network create a highly dynamic network topology that must adapt to the continuous joining or leaving of devices. Within an ad hoc mobile network, mobile devices can collaboratively form clouds that serve networking, storage, and computing purposes.

This variant, known as Mobile Ad Hoc Cloud Computing (MACC) [288], finds application in diverse scenarios such as disaster relief, group live video streaming, and unmanned vehicular systems. MACC harnesses the collaborative potential of ad hoc mobile networks to create transient clouds, offering a decentralized yet resourceful computing environment for dynamic and resource-constrained scenarios.

Mobile Ad Hoc Cloud Computing (MACC) stands apart from traditional cloud computing, primarily owing to the ad hoc nature of its resources. In MACC, mobile devices serve as versatile entities, acting as data providers, storage units, and processing devices. Notably, these mobile devices within a mobile ad hoc cloud network take on the additional responsibility of routing traffic among themselves due to the absence of a dedicated network infrastructure. By collaboratively pooling local mobile resources to create an ad hoc cloud, MACC provides a reasonably high level of computation. This distinctive approach differs from the user focus, architectural principles, and connectivity models prevalent in conventional cloud computing. MACC and MCC differ in hardware utilization, service access methods, and user proximity. In MACC, computation takes place directly on mobile devices, contrasting with MCC where computation occurs at a distance from mobile devices. MACC operates solely on mobile devices, while MCC necessitates the presence of large-scale data centers for cloud computing, alongside mobile devices. This distinction results in MACC offering robust computation power but potentially facing higher latency challenges in MCC. Security considerations in MACC are confined to mobile devices, whereas in MCC, ensuring trust may pose challenges without a secure collaboration framework. Lastly, in MACC, services are exclusively accessed through connected mobile devices using technologies such as Bluetooth, WiFi, and other cellular protocols. This underscores the intrinsic differences in the hardware architecture, security implementation, and service accessibility methods between these two computing paradigms.

While fog computing exhibits versatility across a spectrum of devices, Mobile Ad Hoc Cloud Computing (MACC) excels in scenarios characterized by highly decentralized and dynamic network topologies where a reliable Internet connection is not assured. The decentralized nature of connected devices in MACC surpasses that of fog computing, enabling the formation of a more dynamic network. This is particularly advantageous in environments with sparsely connected devices or networks that undergo constant changes. An illustrative example of this dynamic capability is evident in ad hoc networks designed for peer-to-peer file sharing [127]. In such instances, the inherent decentralization and adaptability of MACC prove advantageous, making it a suitable choice for scenarios where network connections are unpredictable and subject to frequent changes.

Mobile Ad Hoc Networks

Mobile Ad Hoc Networks (MANETs) [15, 29] are comprised of mobile host devices connected to each other in a single hop without the need for base stations [195]. Mobile Ad Hoc Networks (MANETs) are characterized by the presence of mobile nodes that have the freedom to move in and out of the network. The nodes, which can be devices such as cell phones, laptop computers, personal electronic gadgets, MP3 players, and PCs, constitute the network and possess mobility. These nodes are versatile, functioning as hosts, routers, or both simultaneously. They have the ability to establish connections with each other, forming diverse and ad hoc network topologies. One key feature of MANET nodes is their self-configuring capability, allowing them to set up connections swiftly without the need for any pre-existing infrastructure [256]. This self-configuring nature enhances the adaptability of MANETs, making them suitable for dynamic environments where the network topology may change frequently and where the establishment of traditional infrastructure is impractical or unavailable. Unlike MANETs, which form dynamic networks without necessarily constituting a cloud, Mobile Ad Hoc Cloud Computing (MACC) involves the collaborative pooling of computing and storage resources among mobile devices. While MANETs do not inherently form resource pools, various solutions developed for MANETs, such as redundancy and broadcasting, can be adapted and applied to MACC. In resource-constrained environments, peers within MACC may find it beneficial to pool their resources, especially when faced with computationally demanding tasks that may exceed the capabilities of a single mobile device. An illustrative use case for this collaborative resource pooling is seen in unmanned vehicular systems comprising multiple unmanned vehicles and traffic devices.

Multi-Access Edge Computing

Mobile Cloud Computing (MCC) represents an expansion of mobile computing leveraging the capabilities of cloud computing. Similarly, Multi-Access Edge Computing (MEC) serves as an extension of mobile computing through edge computing. According to the definition by the European Telecommunications Standards Institute (ETSI), MEC is a platform that delivers IT and cloud-computing capabilities within the Radio Access Network (RAN) in 4G and 5G, situated in close proximity to mobile subscribers [101]. While initially termed "mobile edge computing," MEC has evolved to encompass a broader spectrum of applications beyond tasks specific to mobile devices. Illustrative examples of MEC applications include video analytics, connected vehicles, health monitoring, and augmented reality. MEC enhances edge computing by furnishing computing and storage resources in the vicinity of low-energy, resource-constrained mobile devices. This allows RAN operators to integrate edge computing functionality seamlessly into existing base stations. Similar to edge computing, MEC can employ small-scale data centers with virtualization capabilities. The computing resources available in MEC, influenced by its underlying hardware, are moderate compared to the scale of cloud computing. Additionally, MEC supports low-latency applications, and its real-time access to radio and network information enables the delivery of personalized and contextualized experiences to mobile subscribers.

Both edge computing and Multi-Access Edge Computing (MEC) services operate at the edge of the Internet, functioning seamlessly even with limited or no Internet connectivity. MEC, however, establishes connectivity through a Wide Area Network (WAN), WiFi, and cellular connections, while edge computing can generally establish connectivity through various means such as Local Area Network (LAN), WiFi, and cellular networks. The operational focus of MEC differs notably from Mobile Cloud Computing (MCC) research. MCC research predominantly explores the dynamics between cloud service users (on mobile devices) and cloud service providers, while MEC research centers around the infrastructure provided by Radio Access Network (RAN) operators. The anticipated integration of MEC with the emerging 5G platform is expected to yield significant benefits [123]. In turn, 5G is perceived as a facilitator for MEC, offering lower latency, higher bandwidth among mobile devices, and extensive support for diverse mobile devices with finer granularity. MEC enhances accessibility to edge computing for a broad spectrum of mobile devices, ensuring reduced latency and more efficient mobile core networks [251]. It facilitates the deployment of mission-critical, delay-sensitive applications over the mobile network [123]. MEC integrates Software-Defined Networking (SDN) and Network Function Virtualization (NFV) capabilities alongside 5G technologies. SDN enables the efficient management of virtual networking devices through software APIs [137], while NFV contributes to reduced deployment times for networking services through virtualized infrastructure. Through the incorporation of SDN and NFV, network engineers, and potentially enterprise application developers, can devise their orchestrator, aimed at coordinating resource provisioning across multiple layers [187].

Cloudlet Computing

Proposed by Carnegie Mellon University, cloudlet computing represents a distinct direction in mobile computing that shares similarities with both Mobile Cloud Computing (MCC) and Multi-Access Edge Computing (MEC), while also addressing some of the limitations of MCC. A cloudlet is essentially a trusted, resource-rich computer or a cluster of computers with a robust Internet connection, strategically positioned to serve nearby mobile devices [228]. Operating as small-scale data centers, akin to miniature clouds, cloudlets are typically positioned just one hop away from mobile devices. The core concept behind cloudlet computing is to offload computational tasks from mobile devices to Virtual Machine (VM)-based cloudlets located at the edge of the network [110]. Although current research in cloudlet computing is predominantly led by academia, it holds significant potential for applications in domains such as wearable cognitive assistance and web applications. Cloudlet computing can be envisioned as the middle tier within a three-tier continuum: mobile device-cloudlet-cloud. Given their nature as small clouds in close proximity to mobile devices, cloudlet computing could be orchestrated by cloud service providers seeking to make their services more accessible to mobile devices. Network infrastructure owners, like AT&T or Nokia, can empower cloudlets with virtualization capabilities, situate them closer to mobile devices, and do so with smaller hardware footprints compared to the extensive data centers employed in traditional cloud computing. While cloudlets may offer more moderate computing resources due to their smaller footprint, they bring about lower latency and energy consumption when compared to traditional cloud computing. The focus of cloudlet computing is to cater to devices in the local area, offering a viable solution for specific use cases.

Mist Computing

Mist computing emerged in the past as addition to the computing landscape, targeting the extreme edge of connected devices – the endpoints. This paradigm encompasses decentralized computing at the most extreme edge, directly involving the Internet of Things (IoT) devices themselves, and is designed with future self-aware and autonomic systems in mind [64, 205]. Positioned as the initial computing layer in the IoT-fog-cloud continuum, mist computing is often informally referred to as "IoT computing" or "things computing". IoT devices within the realm of mist computing can range from wearables and mobile devices to smartwatches and smart fridges. Unlike Mobile Ad Hoc Cloud Computing (MACC), mist computing extends computation, storage, and networking across the fog through the IoT devices. Essentially, mist computing acts as a superset of MACC, encompassing scenarios where networking is not necessarily ad hoc, and devices may not be mobile device. In [241], researchers propose the utilization of nearby mobile devices as a cloud computing environment for storage, caching, and computing purposes. The focus is on reducing the load on traditional WiFi infrastructures for video dissemination applications. The study involves spectators at a sports event organizing themselves into WiFi-Direct groups, exchanging video replays locally, and bypassing the central server and access points. This study exemplifies mist computing, where IoT devices not only function as "thin clients" but also as "thin servers". Other applications of mist computing include preserving user data privacy through local processing [222] and efficiently deploying virtualized instances on single-board computers [190].

Cloud of Things

Another concept closely related to mist computing is the Cloud of Things (CoT) [7], in which IoT devices collectively form a virtualized cloud infrastructure. While mist computing involves computation directly on IoT devices, possibly through message exchange and not necessarily in a pooled resource cloud, Cloud of Things focuses on performing computation over a cloud created by pooling the resources of IoT devices. Abdelwahab et al. [7] introduce the idea of Cloud of Things as a service for sensing, utilizing edge nodes as cloud agents situated in proximity to IoT nodes. The proposal involves dynamically scaling existing cloud resources (compute, storage, and network) by leveraging the sensing capabilities of IoT devices. Edge nodes act as cloud agents near the edge to discover, virtualize, and establish a cloud network of IoT devices (CoT). This network forms a geographically distributed infrastructure, with cloud agents continuously discovering resources of IoT devices and aggregating them as cloud resources. CoT facilitates remote sensing and in-network distributed processing of data. For example, a cloud user could monitor pollution levels in cities based on real-time data from temperature and CO2 concentration sensors in vehicles, ensuring defined accuracy. The CoT framework is scalable to IoT networks, supports heterogeneity among IoT devices and edge computing nodes, and serves as the foundation for sensing-as-a-service using fog computing.

In addition to CoT, another concept presented in [134] is PClouds (personal clouds), which encompasses distributed networked resources derived from both local/personal and remote/public devices and machines. PClouds aim to serve end users even in scenarios where remote cloud resources are either absent or challenging to access due to insufficient network connectivity. An innovative idea akin to Cloud of Things and MACC is introduced in [226], where the authors propose Cloudrone. This concept involves deploying ad hoc micro cloud infrastructures in the sky using low-cost drones, single-board computers, and lightweight OS virtualization technologies. The drones form a cloud computing cluster in the sky, provisioning cloud services closer to the user even in the absence of terrestrial infrastructure to access remote clouds. Similar to Cloud of Things, the concept of Femtoclouds has been introduced to leverage the computational capabilities and pervasiveness of underutilized mobile devices. Femtoclouds utilize clusters of devices often co-located in places like schools, public transit, or malls. Finally, [109] proposes a hybrid edge-cloud workload management scheme for the efficient management of resources and tasks in femtoclouds, aiming to provide low-latency services.

2.1.6 Digital Twins for Frugal and Efficient Industrial IoT Networking

The adoption of the Internet of Things (IoT) is currently spreading in industrial environments. Complex Industrial IoT (IIoT) applications stem from the joint exploitation of multiple and heterogeneous devices adopting multiple protocols and data formats both co-located in the same plant and remotely spread in different locations. Such devices need to coordinate their activities to support industrial operations (e.g., manufacturing and assembly) with stringent quality of service (QoS) and safety critical requirements. The combination of IoT with the industrial ecosystem has recently revitalized the concept of digital twin (DT) as an innovative technology expected to transform industrial and manufacturing ecosystems. Promising improvements are expected to offer new innovative solutions reducing costs, monitoring assets, optimizing maintenance, reducing downtime, and enabling the creation of intelligent connected products. As stated in [108], a DT can be defined as a comprehensive software representation of an individual physical device including its properties, conditions, and behavior throughout the life-cycle of the object. These novel twin-oriented manufacturing systems are characterized by the possibility of supporting and handling the massive heterogeneity of siloed distributed implementations together with protocols and data flows originating from different manufacturing and enterprise services [108, 206].

In this challenging context, network heterogeneity represents a critical element to efficiently handle complex industrial environments and may also significantly limit the design and deployment of Distributed computing applications. DTs, consumers, and services should be unaware of the complexity behind their communications and should be resilient to re-configuration and dynamic orchestration. The layering and separation of functionalities represent a key element, 1) to decouple the networking infrastructure from upper layers and; 2) to dynamically control the communications according to applications and context requirements, e.g., the creation of a segregated and secured network shared only by a group of selected DTs and target data consumers.

By allowing real time monitoring of underlying resources, DTs enable proactive intervention on manufacturing facilities, optimizing interventions and maintenance costs. By doing so, resource-constrained devices can be employed in very simple and with low computational overhead tasks further expanding the applicability of frugality in distributed computing scenarios. Even mobile devices with limited communication and computation capabilities can serve as sensors and actuators because collected metrics help the DT to map the real world state of the systems and allows real time interventions. Anomaly detection [50] within this context exemplifies DT-enabled frugality, with the aim of minimizing machinery replacement costs and reducing infrastructure downtime.

Digital Twins in Frugality Based Applications: SOTA Approaches and Applications

The DT research area is attracting a wide interest involving a multitude of diverse and approaches related, e.g., to big data analytics [218], behavioral modeling [244], ontology definition [249], specific device mirroring [246]. In this fragmented context, the industrial world and in particular the Industrial Internet of Things Consortium is proposing a shared reference architecture [168, 248] taking into account DT relationships, composition, and main services (e.g., prediction, maintenance, safety). It also covers different production stages and use cases, in particular related to manufacturing [146] and product design [264]. However, the limitations of proposed solutions are mainly related to the adoption of a centralized DT management (often deployed only on Cloud infrastructure) where a unique entry point is responsible to maintain twin instances and by moving the integration responsibilities to external modules or connectors. Furthermore, DTs are not in charge of supporting or handling the heterogeneity associated with the connected devices. This missing role feeds the creation of unnecessary substrates of domain specific technologies and legacy interaction forms. In this context, edge processing has already shown its fundamental role to effectively and efficiently handle IoT heterogeneity through the introduction of intermediate proxies and hubs, responsible to manage objects and data streams through centralized approaches. Despite these advancements, the adoption of DTs on the edge is still under-explored and represents a novel research area, including the seamless integration of data and services in heterogeneous systems. Authors in [88] describe the importance of bringing DTs on the edge by highlighting the relevant interest in this new research field and its experimentation. In [58] the authors present two interesting initial evaluations of edge DTs in the specific contexts of blockchain technologies and the social internet of things (SIoT). In [165], instead, the authors propose a different and significant point of view by incorporating DTs into edge networks to support real-time federated learning with reduced communication costs.

In this context, SDN has emerged as the key technology for dynamically managing network configurations in response to real-time insights provided by DTs. Considering fog and edge computing, [197] proposes to exploit SDN to deliver and deploy new services in IoT environments in a faster and more cost-effective manner. The SDN approach can be also adopted together with Blockchain, e.g., to deliver a fully distributed Cloud architecture based on Fog nodes [236] or to improve the credibility and authenticity of nodes while addressing the issues associated with the fact that the SDN controller represents a single point of attack [94]. Finally, SDN is fruitfully adopted to support load balancing in Fog environments.

By focusing on industrial environments, SDN has emerged in the communication research and industrial fields of IIoT [265] primarily to manage switches of closed environments such as datacenters and department networks via the OpenFlow protocol. More recently, [155] focused on the adoption of the SDN paradigm in the context of IIoT to dispatch packets with different delay constraints in a per-flow tailored manner, by considering time deadlines, traffic load balances, and energy consumption. Similarly, [140] adopted SDN to efficiently manage the interplay between edge and cloud environments by considering energy efficiency, bandwidth, and latency. Finally, some efforts have focused on edge IoT scenarios with industrial wireless sensor networks. For instance, [30] manages both transmission scheduling and node mobility allowing to ensure bounded end-to-end delays. [258] aims at improving industrial performance by adopting end-to-end QoS control. To this purpose, it adopts a unique SDN instance for IoT environments consisting of wireless and wired segments, by exploiting 6TiSCH as industrial IoT and open network platform allowing to orchestrate every network segment. Compared with this previous work, our solution adopts SDN to dynamically exploit the communication mechanisms most suitable to current application requirements, ranging from native IP to more articulated ones based on packet content [28]. In addition, it takes advantage of IETF Sensor Measurement Lists (SenML) data format [135] enriched packet payload to efficiently enforce fine-grained content-based traffic flow rules, allowing to better satisfy per-application QoS requirements.

2.1.7 Supporting Fog and Edge Computing with Artificial Intelligence

Managing resources in computing, particularly in the context of Fog and Edge computing, presents a complex challenge due to factors such as resource limitations, heterogeneity, dynamic workloads, and the unpredictable nature of these environments. In addressing this challenge, a growing trend towards leveraging Artificial Intelligence (AI) and Machine Learning (ML) solutions arises. AI/ML methods, particularly those capable of making semi-autonomous decisions like reinforcement learning, appear promising for tackling resource management in Fog computing. However, these algorithms bring their own set of challenges, including high variance [37], issues related to explainability [41], and the need for continual learning [260]. The ever-changing dynamics of Fog and Edge environments demand adaptive solutions that can learn in real-time, adjusting to the evolving computing landscape.

Container orchestration systems are increasingly turning to Machine Learning algorithms to model and predict the behavior of multi-dimensional performance metrics. By harnessing these insights, there is potential to enhance the quality of resource provisioning decisions, particularly in response to the evolving workloads in complex environments. The cloud computing landscape, characterized by diverse and dynamic workloads in large-scale systems, poses a challenge for automating the orchestration process, especially for complex and heterogeneous workloads. Unlike traditional cloud computing platforms, where heuristic policies are common in container orchestrators, there is a growing need to optimize these policies considering the diversity of workload scenarios and Quality of Service (QoS) requirements. As application management in cloud platforms becomes more complex, cloud service providers are increasingly motivated to optimize container orchestration policies by integrating machine learning techniques [281]. A machine learning-based optimization engine employs machine learning models to characterize workloads and analyze performance. It utilizes monitoring data and system logs obtained from the orchestrator to build these models. The engine can then leverage the generated behavior models and prediction results to make future resource provisioning decisions. It may be integrated into the orchestrator or exist independently. The key components of this optimization engine include:

- **Workload Modeler:** This component is specifically crafted for machine learning-based workload characterization. It meticulously analyzes input application workloads, identifying their essential characteristics to inform subsequent modeling;
- **Performance Analyzer:** Utilizing machine learning algorithms, this component generates comprehensive behavior models for both applications and the system. It achieves this by processing monitoring data from the Orchestrator, encompassing both application and infrastructure levels;
- **Predictor:** The Predictor component forecasts workload volumes or application/system behaviors based on the models derived from the Workload Modeler and Performance Analyzer. The prediction results can be seamlessly transmitted to either the Orchestrator or the Decision Maker;
- **Decision Maker:** This component amalgamates the behavior models and prediction results received from the aforementioned components. It employs specific machine learning-based optimization methods/schemes to generate precise resource provisioning decisions, which are then fed back into the Orchestrator.

AI-enhanced orchestration: SOTA Approaches and Applications

In 2016, resource utilization prediction for containerized applications saw the application of ARIMA [185] and nearest neighbor (NN) [291] algorithms. ARIMA, a dynamic stochastic process developed in the 1970s, excels at forecasting non-stationary time series by discerning seasonal differences. On the other hand, NN operates as a proximity search method, identifying the candidate closest to a given point. Widely used in time series prediction, both ARIMA and NN found their initial application in container orchestration, specifically predicting resource utilization metrics like CPU, memory, and I/O. However, during this phase, the application models were relatively straightforward, focusing solely on the time series patterns of infrastructure-level resource metrics.

In 2017, Shah et al. [233] pioneered the application of the long short-term memory (LSTM) model for dependency analysis of microservices. LSTM, rooted in neural network principles, excels in classifying, processing, and forecasting time series data. Distinguishing itself from traditional feedforward neural networks, LSTM incorporates feedback connections to enhance its performance, proving effective in applications such as handwriting and speech recognition. The model proposed assessed both the internal connections among microservice units and the time series patterns of resource metrics. Additionally, anomaly detection was integrated into the LSTM model to identify abnormal behaviors in resource utilization or application performance. In a parallel effort, Cheng et al. [54] leveraged Gradient Boosting Regression (GBR), capable of ensembling multiple weak prediction models (e.g., regression trees) to create a more robust model. They applied GBR for predicting resource demand in workload characterization.

In the pursuit of scaling microservices, Xu et al. [282] employed a model-free Reinforcement Learning (RL) method, specifically Q-Learning. Q-Learning involves learning the action-value function to evaluate the reward associated with taking an action in a given state. One notable advantage of Q-Learning is its ability to achieve the expected reward without relying on a predefined model of the environment. In this context, Xu et al. utilized Q-Learning to generate vertical scaling plans. The objective was to make optimal scaling decisions that minimize resource wastage and computation costs, all while ensuring Service Level Agreement (SLA) assurance.

In the realm of resource provisioning, Deep Reinforcement Learning (DRL) found its initial application in the domain of task scheduling. Bao et al. [21] developed a DRL framework for batch processing job placement, utilizing an Artificial Neural Network (ANN) model to capture the mapping relationship between workload features, system states, and corresponding job placement decisions. The Actor-Critic RL algorithm was employed to train the ANN model, generating optimal scheduling decisions that minimized performance interference between co-located batch jobs. Their solution, compared to traditional heuristic scheduling policies like bin packing, exhibited significant performance improvement on a Kubernetes cluster, particularly in terms of overall job execution time. Furthermore, DRL was also utilized to address the computation offloading problem in fog-cloud environments in Reference [252]. Building upon a Markov Decision Process (MDP) model simulating the interactions of the offloading process at scale, the deep Q-Learning method optimized migration decisions by minimizing time overhead, energy usage, and computational costs. To explore the efficiency of hybrid scaling mechanisms, Rossi et al. [219, 220] leveraged model-based RL models to compose a mixture of horizontal and vertical scaling operations for monolithic applications, aiming to minimize resource usage, performance degradation, and adaptation costs.

In 2020, researchers proposed several Reinforcement Learning (RL)-based scaling approaches, incorporating hybrid Machine Learning (ML) models. Qiu et al. [207] applied the Support Vector Machine (SVM) model for the analysis of microservices dependencies, identifying key components prone to resource bottlenecks and performance degradation. To prevent severe violations of service level objectives (SLO), they employed the Actor-Critic method to make optimal resource assignment decisions through horizontal scaling for these identified components. The approach was implemented and validated on a Kubernetes cluster, demonstrating significant performance improvements compared to Kubernetes's autoscaling approach. Additionally, Sami et al. [223] fused Markov Decision Process (MDP) and SARSA models to develop a horizontal scaling solution for monolithic applications within fog-cloud environments. SARSA generated optimized scaling decisions through model training based on the MDP model, simulating scaling scenarios with fluctuating workloads and considering resource availability in fog environments.

After exploring various computing strategies and providing an overview of intelligent Digital Twins in the Industrial IoT sector, the subsequent sections propose two different middlewares. The first one focuses on the transparent management of distributed network resources through edge located DTs, while the second one illustrates the beneficial application of AI-based packet routing provided by our SDN orchestrator.

2.2 Application-Driven DT Networking (ADTN) Middleware

[26] introduced the ADTN middleware, which employs semantically enriched SDTs on edge nodes and centralized CDTs for flexible orchestration, aiming to reduce management complexity in heterogeneous industrial environments. The middleware utilizes an application-aware SDN solution with multiple traffic forwarding techniques to optimize the tradeoff between packet expressiveness and efficient dispatching. While the proposed solution shows feasibility and efficiency, the article suggests further investigations into providing a wide set of preconfigured SDTs for various industrial protocols, developing CDTs based on high-level representations, and addressing challenges related to enforcing network rules in the presence of competing industrial applications. We designed, developed, and experimentally evaluated the original ADTN middleware that supports the dynamic aggregation and configuration of heterogeneous and sparse industrial equipment, represented as a single business unit. In other words, products and services are, respectively, crafted and supported by an aggregation of things, e.g., sensors, actuators, and simple devices, and their dynamic and flexible orchestration is optimized by an SDN-based cross-layer approach taking into consideration application-driven indications together with QoS requirements and network configuration adaptation capabilities. The ADTN middleware is responsible to handle and effectively orchestrate scalable and reliable communications among physical devices, DTs, and modules with respect to a dynamic set of application-driven rules and indications coming from external authorized services. In particular, primary ADTN components are the simple digital twin (SDT) on the edge side and the composed digital twin (CDT) on the CR side (see Fig. 2.4). SDT is a software agent running at the edge layer providing an effective one-to-one mirroring of a physical IoT device through the digitalization and cloning of all its features and functionalities. Each SDT maintains the communication and synchronization with the associated counterpart creating a standardized and

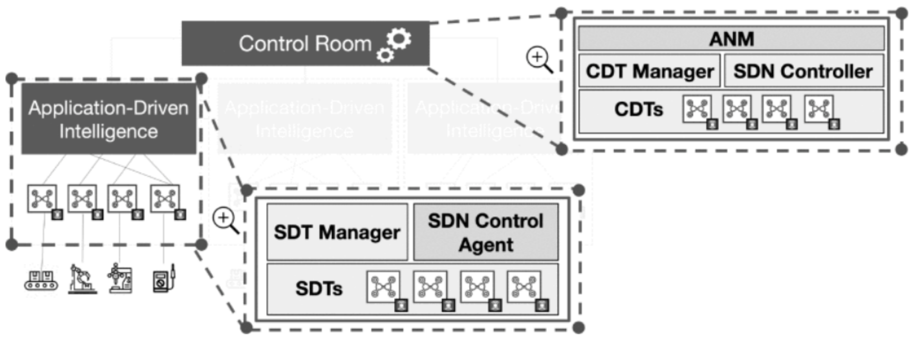


Figure 2.4: Primary components of the ADTN middleware.

uniform abstraction of the device to enable interoperability and cooperation of devices, services, and applications. Furthermore, the SDT can extend the original device's behavior in terms of supported protocols, integrating with external services, and managing how incoming and outgoing packets are exchanged and internally processed. In other words, the SDT provides the opportunity of dynamically augmenting the information coming from physical things through the conveniently reformatting/pre-processing of headers and payloads or the introduction of additional metadata.

CDT is a software component in the CR layer shaping the digital representation of a new entity capable to aggregate multiple SDTs at the same time to efficiently model complex distributed applications and behaviors. In real deployment environments, physical devices are often a composition of different heterogeneous components, and things from different edge locations can participate in common application goals, such as the various tools in the smartphone assembly example. The ability - denoted as Composability - of grouping different objects into an aggregated one and then observing and controlling the behavior of the resulting object (as well as the individual entities) represents a strategic feature for DT design and development. Thanks to the standardization and homogeneity obtained through the use of SDTs, a CDT has the ability to easily compose and aggregate multiple twins while abstracting the complexity of a larger system and focusing only on a few application-oriented relevant status and behaviors. In addition to SDTs, the edge layer is composed of:

- **STD-Manager (SDT-M):** with the responsibility of configuring, instantiating, and handling the life-cycle of SDTs. Each edge node hosts an independent SDT-M associated with the middleware to create and maintain an active virtual replica for physical devices the edge node is directly connected to. The SDT-M performs SDT advertising and receives SDT configuration commands by remotely interacting with the CDT Manager;

- **SDN Control Agent (S-CA):** residing on host nodes together with the SDT-M. S-CA remotely interacts with the CR (and in particular the SDN Controller, see below) to send information about edge node computational, memory, networking characteristics, and its current state. Moreover, S-CA receives from the CR the traffic engineering rules to apply to the local edge node with the goal of optimizing packet management in an application-driven manner.

In addition to CDTs, the CR layer is composed of:

- **CDT Manager (CDT-M):** a software component orchestrating the creation and management of CDTs, according to application-driven rules and specifications. CDT-M actively communicates with deployed and active S-CAs to gather information about available physical devices and about supported features of related SDTs. The CDT-M is also responsible to push and dynamically adapt CDTs' configurations to shape their behavior and how they can react to context variations detected by active SDTs;

- **SDN Controller (S-Ctrl):** which i) receives per-edge node networking data from S-CAs, with the goal of generating the whole network topology, and ii) remotely distributes and activates traffic engineering rules, based on CDT QoS requirements.;
- **Application-driven Network Manager (ANM):** the entry point of the whole solution allowing technicians to add/remove and de/activate CDTs (via CDT-M) and to manage the whole topology (via S-Ctrl). To this purpose, ANM actively interacts with the local CDT-M to get the set of running CDTs and related QoS requirements and with the local S-Ctrl to manage remote edge nodes and require them to properly tune network resources to activate traffic management rules, e.g., to achieve the desired delay, jitter, and/or throughput.

Fine-Grained Application-Driven QoS Management

Each SDT hosted on an edge node is uniquely identifiable and reachable through a dedicated logical name or a distinctive identifier, e.g., logic-name:port such AS:1234), able to hide its actual IP address, thus allowing to route traffic flows and packets while reducing networking complexity. To this purpose, the developed ADTN middleware exploits the SDN-based multilayer routing (MLR) approach, specifically supporting network management in edge-based multihop deployment environments [28]. MLR allows to exploit, even at the same time, different routing strategies and mechanisms suitable for applications with heterogeneous features and requirements. Based on its centralized point of view, S-Ctrl dynamically determines and configures the proper MLR forwarding mechanism, ranging from traditional IP and sequence-based overlays to more articulated forwarding solutions based on the inspection of payload content types and values. In particular, the SDN-enabled MLR approach allows to exploit the same network topology even at the same time by different industrial applications (represented by different CDTs) to dispatch traffic flows based on native IP, overlay networking information, and payload content. Moreover, MLR allows the exploitation of the multihop and multipath network by supporting an overlay network that distinguishes edge nodes based on fixed unique identifiers rather than with time-varying private IP addresses.

S-Ctrl adopts four different network management approaches, by also considering the payload type. To this purpose, S-Ctrl and S-CAs identify three packet types: Video, carrying frames provided by surveillance cameras, Vibration, representing the data provided by the drill, and Info, logs generated by the conveyor, the assembler, and the checker. In case there is no network congestion, network management rules are not activated and thus packets are dispatched in a best effort manner. In case S-Ctrl identifies low network congestion, S-Ctrl enables on S-CAs a network management rule dropping packets with probability $(percentage/100)^{hopCount}$ (with hopCount > 0) and percentage set to 25%, 33%, and 33% for Info, Video, and Vibration packets, respectively. Of course, this rule only applies to packets tagged as droppable. For instance, according to the Smartphone Assembly policies and Listings 1 and 2, Vibration packets can be dropped only in case no anomalies have been detected. Let us note that by dropping packets based on the hop count, the probability of dropping a packet along a multihop path is

$$\sum_{hopCount=1}^{pathLength} \left(\frac{percentage}{100}\right)^{hopCount} \tag{2.2}$$

and with $pathlength \to \infty$ the overall probability a packet is dropped is 33% and 50% if *percentage* is set to 25% and 33% respectively. In this manner we achieve the notable effect of adopting a simple dropping mechanism that can be applied on each edge node in a stateless manner, while imposing a limit to the percentage of packets that can be dropped.

Experimental setup and Evaluation

Achieved performance results are based on our working Java prototype of the ADTN middleware that we have developed not only to demonstrate the feasibility and the efficiency of the presented model, but also to provide the community with a working solution to foster the research in this field. The source code of the adopted libraries and implementations has been released as Open Source project[1]. In particular, we analyzed two different aspects. The first one relates to the investigation of SDT performance in terms of i) SenML data enrichment delay and ii) its impact on the twin forwarding delay between data producers and consumers. The second aspect is about how our ADTN middleware dynamically manages network configurations in relation to application requirements and network state.

In the former case, involved tests have been conducted on ten independent runs considering an uncongested network, a target set of 10000 messages with IoT smart objects and DTs using MQTT as protocol, and an average payload size of 100 bytes. SDTs are implemented using Java and the WLDT library, a modular software stack based on a shared multi-thread engine able to effectively implement DT behaviour and to define its mirroring procedures, data

[1]https://github.com/DSG-UniFE/ramp

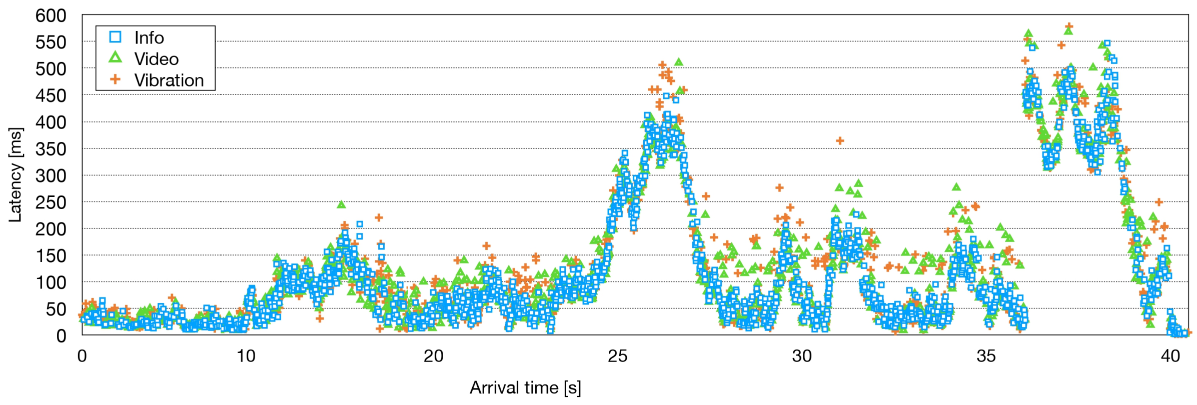


Figure 2.5: Message delay at increasing message rate while applying the proposed QoS management solution.

processing, and the interaction with external applications. Implemented SDTs are executed as independent processes, but can also be easily packed as containers to run on virtualized environments and microservices. In order to evaluate the SDT's suitability to different hardware profiles, we tested the implementation both on an high specs Linux edge node (i7 Intel CPU and 32 GB of RAM) and on a Raspberry Pi (RPi) Model B board with 700 MHz CPU, 128 MB of RAM and a 10/100 Mbps Ethernet connectivity.

In the latter case, we focused on the capacity of the ADTN middleware to dynamically introduce new networking policies depending on the congestion level of the network, by activating different traffic management rules on an intermediary edge node To this purpose we made measurements over a test-bed composed by N1, N2, and N3, three high specs RPis (Model 3B+ with 1 Gb RAM and 1.4GHz 64-bit quad-core processor). N1 and N2 are connected via 100 Mbps Ethernet, N2 and N3 via 10 Mbps IEEE 802.11. N1 acts as SDT and sends data (with 100 bytes payload), while N3 is the receiver and N2 plays the role of the intermediary edge node. At the beginning, N2 is configured with *no network congestion* rule and N1 sends data at 150 msg/sec. Then, N1 increases the message rate by 500 msg/sec every 12 s. As Fig. 2.5 shows, at about 12 s, the receiver notices a packet latency greater than the 150 ms threshold ms and informs S-Ctrl about it. Then, S-Ctrl informs S-CA on N2 to activate the *low network congestion*, i.e., the intermediary edge node has to discard or delay part of Video and Vibration messages. Network congestion lowers for a while and N3 receives packets at reduced latency. However, the message rate keeps increasing and at about 25 s N3 notices a latency greater than 300 ms and again it interacts with S-Ctrl to trigger the activation of the *medium network congestion*. This rule further increases the amount of dropped and delayed packets, limiting the overall packet latency. Finally, at about 36 s the latency becomes greater than 450 ms. The receiver informs S-Ctrl about the current situation and the latter activates the *high network congestion* rule, thus imposing sender and receiver to communicate via OS routing instead of the overlay network. Note that while performing the procedure to switch to OS routing the sender keeps sending packets, thus even after the receiver has notified the high latency to S-Ctrl some packets are still sent via the overlay network, and thus some packets still present high latency. In this case the network latency considerably lowers, but with the drawback of reduced traffic management at the application level, since N2 is not able to access the payload of traversing packets anymore.

Discussion:

The results presented above demonstrate that the ADTN middleware is able, on the one hand, to enrich packets in a very efficient manner also providing a uniformed standard data layer and, on the other hand, to dynamically manage traffic flows in an application-driven manner also considering the current state of the network. In particular, we found that the additional overhead imposed by SDTs to format and enrich the packet payload in the SenML syntax is very limited, largely justified by the advantages of greatly improving the QoS management features of traffic flows via articulated and fine-grained network rules on intermediary edge nodes. However, it is worth noting that while experimental results prove that the ADTN middleware provides an efficient high level interface for DT management, they do not show how our solution behaves in terms of other highly demanding QoS requirements, e.g., scalability, privacy awareness, and reliability. Therefore, based on the encouraging results already obtained, we are now working on the development of a full-fledged industrial pilot.

2.3 ML-based Orchestration for Security and Performance Optimization in Industrial Environments

In contemporary industrial networking environments, integrating Operation Technology and Information Technology necessitates a focus on ensuring both operational safety and security. However, their practical implementation often relies on error-prone human-centric procedures by technicians, lacking an integrated plant-wide perspective. To address these challenges, our approach emphasizes the application of intelligent reasoning to monitor and actively reconfigure the environment, ensuring the fulfillment of security requirements while optimizing the trade-off between security and performance. Our prototype-based performance results demonstrate the feasibility and efficiency of our solution under stringent industrial requirements, showcasing flexibility in creating new configurations and satisfying low-latency requirements. Our prototype-based performance results demonstrate the feasibility and efficiency of our solution under stringent industrial requirements, showcasing flexibility in creating new configurations and satisfying low-latency requirements. In particular, our ML orchestrator proved capable of creating a new configuration in less than 2.5 s on a typical edge node with a medium load.

Secure Intelligent Digital Twins (SIDI) architecture

Our SIDI middleware exploits ZDT and CDT definitions by considering performance in terms of bandwidth and latency as well as security in terms of adopted ciphering algorithm and key length. In addition, we originally differentiate among Network DTs (NDTs) and Asset DTs (ADTs) to provide a virtualized representation of the underlying architecture, with the formers for machines and digital services and the latters for network elements (either physical or virtual) allowing inter-zone communication. The middleware of four primary modules:

- **ADT & NDT Managers (ANM):** it is in charge of handling the creation, coordination, and dynamic management of ADTs and NDTs. ANM instances can be deployed also on connected edge nodes to support DTs remote deployment in case of strict latency constraints or if it is require to locally communicate with edge equipment and applications;

- **Conduit & Zone Manager (CZM):** allowing OT technicians to de/activate CDTs and specify available ZDTs starting from incoming "high level" requirements and commands, e.g., "Create a new conduit between Zone A and Zone B with High QoS level". CZM communicates with deployed ANMs to retrieve information about active DTs and properly populate defined zones or to create conduits according to the target mapped requirements;

- **Metrics Collector:** gathering all the available metrics coming from active components at different abstraction levels. For instance, it receives the status of available computational nodes (e.g., CPU, memory and bandwidth usage), real-time network information coming from NDTs (e.g., delay, throughput, and packet loss), and assets' metrics from ADTs (e.g., requests per second, average response time, and exceptions). All the collected information are used by the RICM (see below) to dynamically manage ADTs and NDTs, thus to enforce provided requirements;

- **Reasoning Engine & Intelligent Context Manager (RICM):** training a Machine Learning model on the data collected by the Metric Collector to obtain a predictor able to forecast the next state of the network taking into account the current one and the possible routes. It is also in charge of selecting routes better satisfying the QoS requirements

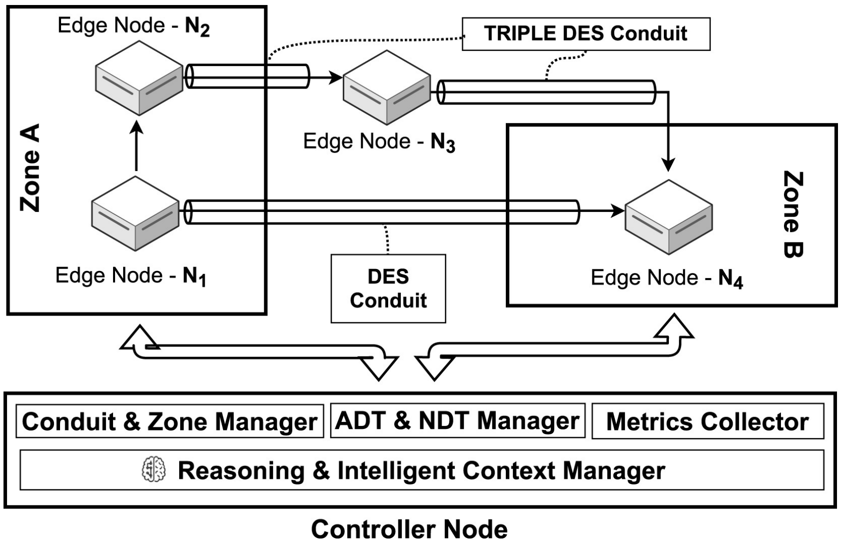


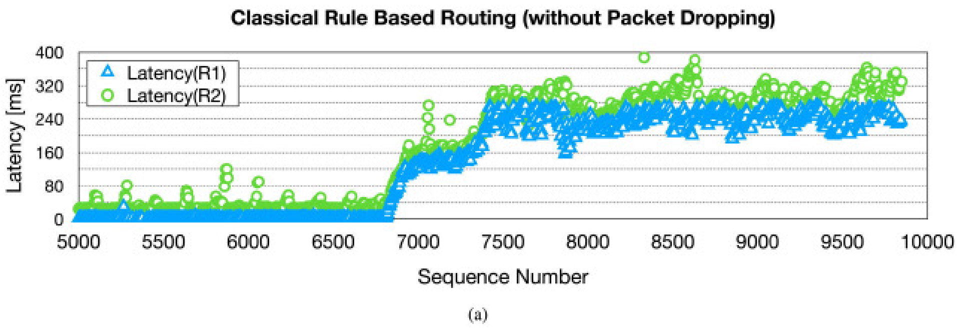
Figure 2.6: SIDI middleware architecture highlighting main components

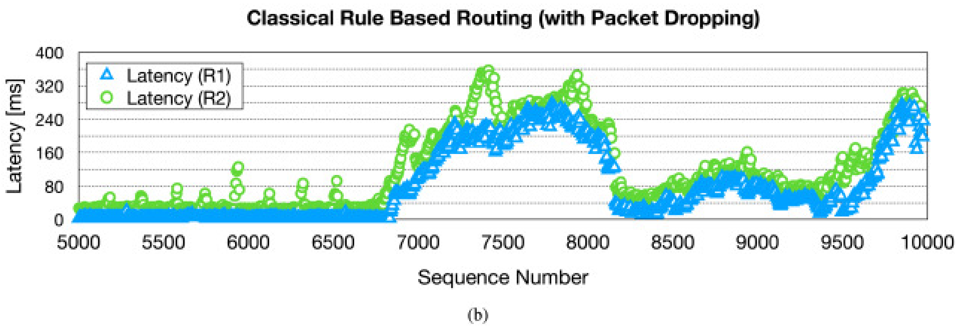
declared at IBN level (see Section 4.2). RICM acts as a predictor for the future behavior of the network and as a dynamic policy manager at the same time. By doing so, it is able to react to unpredictable and undesirable states by itself, without any kind of manual update of human operators.

The SIDI middleware is designed to enhance operational efficiency and scalability, reducing reliance on human intervention and minimizing errors. It facilitates dynamic orchestration and automation, particularly supporting artificial intelligence techniques. The Intelligent Context Manager collaborates with a Metrics Collector and a Reasoning Engine to dynamically forward packets based on conduit policies, learning network behavior and selecting optimal routes for satisfying Quality of Service (QoS) requirements. By collecting real-time metrics and predicting packet latency, the system can proactively avoid undesired congestions and prioritize QoS requirements. The potential extension involves moving certain components to edge nodes for distributed decision-making and delegating resource-intensive operations to a centralized node.

Implementation Insights and Experimental Evaluation

We have proved the feasibility and efficiency of the proposed approach by setting up and implementing a testbed composed of 4 constrained devices. In our testbed, each node hosts a Docker instance and two containers, one holding the application logic and handling the connection and the communication with the other nodes and another one exporting resource usage metrics to the Intelligent Context Manager. For the application level container we choose NodeJs as





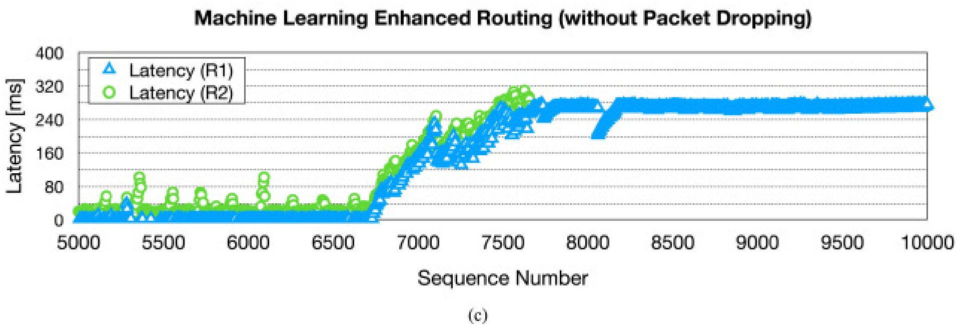


Figure 2.7: (a) Classical rule based routing without packet dropping; (b) Test 2: Classical rule based routing with packet dropping; and (c) Test 3: Machine Learning (ML) enhanced routing.

developing framework since very efficient and lightweight and we adopted Prometheus and Node-exporter for resource monitoring. Prometheus is an open-source monitoring solution for highly dimensional data. It provides high level APIs to obtain the data collected by querying the system with a specific query language called PromQL. Node-exporter is a monitoring tool container which collects resource usage metrics from client nodes in a passive way, without spoiling the data collection process with additional monitoring overhead.

We run three different tests where 30% of packets are sent over route R1 and the remaining 70% over route R2 while the sending frequency constantly increases from 100 pkt/s to 1000 pkt/s. During the first test, no latency saving policies are applied, while the following two tests have been done with a maximum latency requirement equal to 300 ms. As a consequence, Fig. 2.7(a) shows as the maximum latency reaches 400 ms without any kind of reaction from the Controller Node. Fig. 2.7(b) illustrates a second test where static routing rules are applied to reduce the overall latency. Once latency reaches 300 ms the receiver node drops 35% of traversing packets. As it can be seen, this leads to lower latency for a transitional state, but after that, the network is not able to keep up with the increasing packet frequency even with some artificial packet loss addition. In this case, despite the maximum latency requirement is satisfied for some time, the solution leads to a considerable information loss and it is not robust to the dynamic evolution of the current state of the network. On the other hand, Fig. 2.7(c) shows the results achieved in a third test where the Controller Node benefits from the support of the Reasoning Engine which predicts the expected latency depending on a specific route choice. The Intelligent Context Manager is then able to select the route corresponding to the lower latency value. In this case, when the latency reaches 300 ms the majority of packets is forwarded over R1, prioritizing lower latency over security and leading to an average latency equal to 275 ms, well below the predefined maximum threshold.

Achieved results show how the SIDI middleware (based on a Machine Learning enhanced policy manager) is able to change the network behavior before a real congestion takes place, autonomously. In addition, no compromises in terms of information loss have to be adopted to guarantee the satisfaction of QoS requirements. The third test exploits the predictive capabilities of the Reasoning Engine. It implements a K-Nearest Neighbors ML model trained with a data-set containing five different simulations with 5000 forwarded packets each. The model has been developed with the scikit-learn python framework [200], a well-known package for classical and efficient predictive data analysis. Data coming from both the application container and the resource monitoring container are merged for each node to model the state of each ADT in terms of resource usage, current packet frequency, and corresponding final latency. Thanks to the uniform interface provided by the underlying SIDI architecture the Reasoning Engine can model the future state of the network in a transparent way for the human operator. Furthermore, a new and more complex ML model will benefit from the transparency offered by the networking layer and can be added and compared with minimal additional effort. By delving into finer details, the data-set has been divided into training and test sets to have four simulations for training and the remaining for testing. During the training process, the model has been cross-validated by training on three simulations and validated on the fourth iteratively four times. The resulting model reaches a test accuracy of about 87% and can prioritize the R1 route over R2 successfully. The model evaluation has been done by considering both Rooted Mean Squared Error for hyperparameters validation and Mean Absolute Percentage Error for the final accuracy score. An exhaustive grid search has been used to tune the model by selecting the best k value for the number of neighbors to consider. It identified $k = 3$ as the best value, meaning that to predict the latency related to a single packet the metrics of the closest three packets have to be taken into account.

The SIDI middleware leverages the Digital Twin (DT) approach to create virtualized representations of machines and network elements, facilitating the definition of inter-machine security requirements. The reasoner, based on DTs, dynamically monitors and reconfigures machines and network elements to ensure requirements are met. Performance results from the open-source prototype show feasibility, advantages, and efficiency, but real-world validation in industrial environments is necessary. Future work involves testing in more complex scenarios, integrating with an actual industrial testbed, addressing limitations in the machine learning model, and exploring the trade-offs of using virtualized network functions in industrial environments for enhanced flexibility.

Chapter 3

Energy-Efficient Frugal Approaches for Constrained Devices in Distributed Learning

Edge AI refers to the practical implementation of Artificial Intelligence (AI) in real-world devices, involving the execution of AI computations near users at the network edge rather than in centralized cloud data centers. This chapter offers a comprehensive analysis of AI methodologies and capabilities in the context of edge computing, exploring the transition to Edge AI and providing insights into various paradigms of edge computing. In particular, it delves into the deployment of AI algorithms and models on resource-constrained edge devices, illustrating their applications in IoT scenarios such as autonomous vehicles, smart homes, industrial automation, healthcare, and surveillance. Additionally, the chapter addresses the use of machine learning algorithms optimized for resource-constrained environments, highlighting key challenges and potential research directions for the future development of Edge AI.

Here, the focus is in the implementation of AI models directly on edge devices, considering the execution of both the training and inference phases of AI models, by emphasizing the synergy between devices, edge computing, and the cloud. Related to this chapter, in Section 3.1 we will try to clarify the concept of Edge Intelligence and the main obstacles to its adoption in real-world settings. By introducing greener and more sustainable ways for distributed intelligence we focus our attention on the usage of frugality for energy optimization. In this direction, Section 3.2 and Section 3.3 introduce experimental contributions, incrementally.

3.1 Edge AI: When Intelligence Moves Towards the Edge

Recent advancements in key dimensions have facilitated the successful deployment of AI models at the edge [279]. Key foundations for generalized machine learning have been established through progress in neural networks and various AI domains. Organizations are increasingly recognizing the feasibility of training and operational deployment of AI models at the edge [152]. Massively distributed computing capacity is essential for AI at the edge, utilizing recent developments in massively parallel GPUs for running neural networks [280]. The exponential growth in data volume, propelled by the proliferation of IoT-connected devices [211], has paved the way for implementing AI models across various industries. The widespread availability of sensors, smart cameras, robotics, and other data-gathering tools enables the integration of AI in diverse applications [98]. Furthermore, the enhanced speed, reliability, and security brought by 5G/6G are contributing to the advancement of IoT [9]. According with [68] the main driving indicators in terms of QoS in Edge Intelligence are:

- **Performance:** performance metrics primarily encompass training loss and inference accuracy, pivotal for assessing AI models. Despite the shift in computational scenarios from cloud clusters to a harmonized system involving devices, edge, and cloud, these criteria remain crucial.;

- **Cost:** cost considerations typically include computation cost, communication cost, and energy consumption. Computation cost reflects the demand for computing resources, such as CPU cycle frequency and allocated CPU time. Communication cost involves communication resource requirements like power, frequency band, and access time. Minimizing delay (latency) caused by allocated computation and communication resources is also a focal point. Given the limited battery capacity of mobile devices, efficient energy consumption is vital. Cost reduction is a key goal, with edge computing aiming for substantial reductions in delay and energy consumption, addressing critical challenges for realizing 5G;

- **Privacy:** n the context of heightened concerns about data leaks [177], privacy preservation has gained significant attention. Federated learning, which aggregates local machine learning models from distributed devices while preventing data leakage, has emerged as a solution. Privacy is closely linked with security and is associated with the robustness of middleware and software in edge systems;

- **Efficiency:** high efficiency is crucial for achieving excellent performance with minimal overhead. Improving existing algorithms and models, especially for AI on the edge,

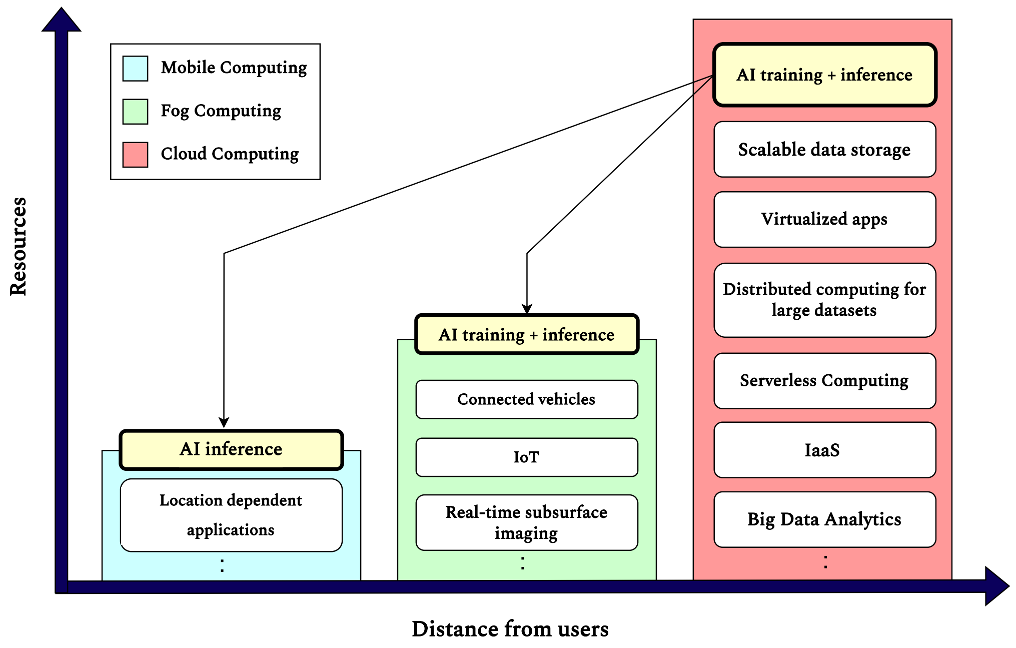


Figure 3.1: The focus on AI-enhanced applications at the Edge of the network.

is driven by the pursuit of efficiency. Approaches such as model compression, conditional computation, and algorithm asynchronization have been proposed to enhance the efficiency of training and inference in deep AI models;

- **Reliability:** system reliability ensures uninterrupted operation throughout prescribed periods, a key determinant of user experience. In the context of edge intelligence, system reliability gains prominence for AI on the edge due to the distributed and synchronized nature of model training and inference. Local users participating in these processes face a significant probability of failure due to issues like wireless network congestion, emphasizing the importance of reliability.

The integration of edge computing and artificial intelligence (AI) has given rise to the concept of edge intelligence, driven by the need for quick analysis of large volumes of data and extracting insights. Currently, there is no universally accepted definition of edge intelligence, leading researchers to propose their own interpretations. For instance, Zhou et al. [296] argue that edge intelligence goes beyond merely running AI models on edge servers or devices; instead, it involves a collaborative approach between edge and cloud computing. They define six levels of edge intelligence, ranging from cloud-edge co-inference (level 1) to full on-device processing (level 6). Another definition by Zhang et al. [293] characterizes edge intelligence as the capability enabling edges to execute AI algorithms.

AI on the Edge: SOTA Approaches and Applications

As mentioned earlier, optimizing communication efficiency is crucial for Federated Learning, especially when transitioning it to the edge. The primary objective is to minimize the number of communication rounds, as updating the global model may face challenges when local devices are offline or the network is congested. Numerous efforts concentrate on reducing communication overhead for Federated Learning, with model compression being a key strategy. Compressing trained models without compromising inference accuracy is a highly effective approach. For instance, in [144], structured updates and sketched updates are proposed to decrease uplink communication costs. Structured updates involve learning the local update from a restricted lower-dimensional space, while sketched updates compress the uploaded model before transmission to the central server.

Bonawitz et al. [35] introduced a communication-efficient secure aggregation protocol specifically tailored for high-dimensional data. The protocol exhibits robustness by tolerating up to 33.3% of participating devices failing to complete the protocol. In a related work, it is noted that Deep Neural Networks (DNNs) are often overparameterized, featuring significant redundancy in their weights. The proposed solution involves pruning, a technique that compensates for the loss in performance. The article suggests a retraining-after-pruning scheme, where the DNN is retrained on new data while the pruned weights remain constant. This scheme effectively reduces resource occupation while maintaining learning accuracy.

In the healthcare sector, IoT solutions have been pivotal in the evolution of health management systems, facilitating efficient tracking of agents such as patients, medical practitioners, and resources. This transformative approach is referred to as the Internet of Health Things (IoHT) [255]. A critical component of IoHT is the utilization of wearable sensors, which collect health-related parameters at various time intervals. These data are then processed to drive the development of intelligent e-healthcare applications. In [128] the resulting health monitoring process involves training with a deep learning network and the elimination of weak classifiers through feature ensemble computation. The system's efficiency is evaluated through MATLAB-based experimental analysis, achieving an accuracy of up to 98.7%. Future enhancements include optimizing techniques and implementing feature selection approaches to further refine the monitoring process.

The work introduced in [74] proposes a novel online approach based on deep learning and transfer learning concepts to create a computational intelligence framework for Internet of Health Things (IoHT) devices. This user-friendly framework enables easy addition of images and platform training, making it accessible even to individuals without programming or image processing expertise. Trials demonstrated rapid project setup by users with no technical background. The proposed approach is validated using three medical databases for stroke type classification, lung nodule malignancy classification, and melanocytic lesion classification. The framework achieves high accuracy, reaching 91.6% accuracy in stroke and lung nodule databases and 92% accuracy in skin images databases. The results highlight the framework's efficiency and reliability, showcasing its potential to assist medical professionals in quickly and accurately analyzing complex medical examinations through a collaborative IoT platform.

In recent years, the integration of Internet of Things (IoT) solutions has transformed traditional homes into smart living spaces, offering a myriad of automated and interconnected services. This subsection explores the innovative applications of IoT in provisioning smart home services, encompassing the automatic control of domestic appliances, alarm generation, security controls, and the development of comprehensive Internet-connected systems. From enhancing convenience to improving security, IoT technologies have reshaped domestic living, paving the way for a more efficient and interconnected future in our homes.

Gavrila et al. [96] introduce an architecture for seamlessly integrating a Smart Home (SH) environment with an HbbTV-enabled television set and consumer's handheld devices, such as smartphones and tablets. Leveraging the HbbTV 2.0.1 Companions Screen and Multimedia Synchronization framework, the proposed solution acts as a central hub for audio and video broadcasting services, home automation, and various devices within the Smart Home. The architecture, designed to be standard-compliant and platform-independent, demonstrates successful hardware implementation and is validated through subjective tests based on mean opinion score (MOS). The results indicate that the accurate design of hybrid services through HbbTV can enhance the Quality of Experience (QoE) perceived by users in the unified TV and Smart Home services.

An intelligent and multi-objective framework for managing residential loads in smart homes is introduced in [51]. This framework utilizes an IoT-based controller to efficiently handle home loads and promptly issues alerts in case of any detected malfunction within the household appliances. Considering the context of smart homes, it acknowledges the potential risks posed by cyberattacks, which not only affect the functionality of the home but also pose threats to the safety and well-being of the occupants.

The integration of IoT has significantly enhanced agricultural production by improving the quality of agricultural products, reducing labor costs, and enabling more efficient farm management practices. Hu et al. [121] address the importance of precise crop disease diagnosis in agricultural research and production. The traditional identification of coarse-grained diseases in crops is deemed insufficient, as treatment methods vary within different grades of the same disease. To address this, the study proposes an IoT system that integrates IoT technology and deep learning for fine-grained disease identification in crops. The system automatically detects diseases and communicates diagnostic results to farmers. The proposed multidimensional feature compensation residual neural network (MDFC-ResNet) model enhances fine-grained disease identification by considering species, coarse-grained disease, and fine-grained disease dimensions. The compensation layer, employing a compensation algorithm, fuses results from multidimensional recognition. Experimental results demonstrate that MDFC-ResNet outperforms other popular deep learning models, providing better recognition accuracy and practical guidance in agricultural production activities.

The study proposed in [295] an autonomous agricultural system to address labor shortages in agriculture while enhancing productivity. The system incorporates a ground-level mapping and navigation system using computer vision (Mesh-SLAM algorithm) and the Internet of Things (IoT). The three-layered system includes robot vehicles for frame collection, edge nodes for image feature data edge computing, and a cloud layer for general management and deep computing. The Mesh-SLAM algorithm efficiently maps the farm in 3D, and the IoT architecture allows scalability and flexibility. Evaluation results indicate superior mapping and localization precision, making the system feasible for practical implementation in real farms by balancing cost and performance.

The Internet of Things (IoT) holds significant potential for Intelligent Transportation Systems (ITS), contributing to the creation of a smart, safe, reliable, and sustainable transportation network. This is achieved through the collection and analysis of traffic and mobility-related data using IoT technologies in ITS [297]. The study conducted by Ke et al. [141]explores the use of edge computing in smart parking surveillance, focusing on parking occupancy detection through real-time video feeds. The system employs a carefully designed processing pipeline, considering flexibility, online surveillance, data transmission, detection accuracy, and system reliability. It incorporates edge-based artificial intelligence, implementing an enhanced single shot multibox detector (SSD) and additional algorithms for optimal efficiency and accuracy. Field tests conducted in an actual parking garage demonstrate promising results, achieving over 95% accuracy in real-world scenarios. The proposed smart parking surveillance system is seen as a crucial element for smart cities and a foundation for future applications in intelligent transportation systems.

Philip et al. [202] address real-time Internet of Things (IoT) applications, specifically focusing on smart traffic control scenarios involving autonomous vehicles making decisions on lane velocities. The study decomposes the problem as an unconstrained network utility maximization problem and proposes a consensus-based, constant step-size gradient descent algorithm to achieve a near-optimal solution. The analysis emphasizes the delay-accuracy trade-off, measuring delay in terms of the number of iterations required for scheduling operations within an acceptable tolerance. The algorithm's operation under quantized message passing is also explored. Simulations, incorporating microscopic traffic flow behavior, compare the proposed solution with traditional and state-of-the-art intersection management techniques.

The Internet of Things (IoT) embodies the idea of everyday objects connecting us to the IoT era. The spatial characteristics of surrounding objects serve as the control mechanism for IoT functionality. Essentially, the spatial changes in an object represent the essence of IoT responsiveness. For instance, variations in crop spatial patterns signify the needs and characteristics of agricultural production. Similarly, the spatial dynamics of human movement can trigger alerts in security and monitoring systems. This consideration prompts the exploration of the "Internet of Spatial Things (IoST)" concept [78, 225]. Ghosh et al. [100] propose a mobility-driven cloud-fog-edge collaborative real-time framework, Mobi-IoST, designed for IoT systems with a back-end cloud. The framework leverages IoT, Edge, Fog, and Cloud layers, treating IoT and edge devices as moving agents in a 2-D space. By analyzing spatio-temporal mobility data and contextual information, Mobi-IoST employs machine learning algorithms for real-time prediction of agent locations. The framework features hierarchical processing, utilizing an IoT-Edge-Fog-Cloud architecture for improved real-time application Quality of Service (QoS). It effectively predicts agent locations, reducing delays and power consumption by approximately 23-26% and 37-41%, respectively, compared to existing mobility-aware task delegation systems, with an accuracy of approximately 93%.

Koh et al. [143] present a novel location spoofing detection algorithm named Enhanced Location Spoofing Detection using Audibility (ELSA) for geo-spatial tagging and location-based services in the Internet of Things (IoT). ELSA operates at the backend server and is designed to be implemented seamlessly with existing IoT systems. Utilizing a statistical decision theory framework and two-way time-of-arrival (TW-TOA) information, ELSA incorporates implicit audibility information to enhance location spoofing attack detection rates. The algorithm employs a generalized likelihood ratio test for verifying device locations and demonstrates superior performance, as evidenced by extensive simulations on synthetic and real-world datasets, outperforming conventional non-audibility-aware approaches in terms of detection and false alarm rates.

Exploring the intricate relationship between AI for/on the Edge, Section 3.1.1 focuses on more sustainable AI based solution and contributions from the current state of the art highlighting the main challenges in terms of performance and optimization tradeoffs.

3.1.1 Green and Frugal AI

AI, already integral to numerous systems, is poised to become even more pervasive with the rapid ascent of wearable devices and the Internet of Things. In traditional machine learning applications, emphasis is placed on achieving high-quality results, necessitating substantial data collection and computational resources for model building. However, in various scenarios, establishing large centralized data repositories is unfeasible or impractical. For instance, in personal health, privacy concerns may impede the sharing of detailed personal data. In these cases, an ideal solution involves performing machine learning directly on wearable devices, introducing significant computational challenges, such as the limitation of smartwatch battery capacity.

Paradoxically, deep learning draws inspiration from the exceptionally energy-efficient human brain. However, the substantial computational costs associated with deep learning pose challenges for academics, students, and researchers, particularly those in emerging economies, limiting their participation in advanced research endeavors. For these reasons, recent studies such as the one conducted in [231], have started questioning the current *Red AI* trend proposing a more *Green AI* paradigm. The term *Red AI* refers to AI research that aims to improve accuracy by using massive computational power, by essentially "buying" stronger results with more resources. In their work, the authors first provide an empirical study showing the prevalence of *Red AI* articles over energy saving approaches, surveying 60 papers from top AI conferences over the period of 2018-2019 period. They report that about 80% of sampled papers targeted accuracy over efficiency, by prioritizing the final result over the potential lower impact of less performing solutions. The mathematical representation of this concept is formulated thought the *Equation of Red AI*, as follows:

$$Cost(R) \quad \alpha \quad E \times D \times H \tag{3.1}$$

where $Cost(R)$ is the cost of a single ML experiment, E is the cost of processing a single example, D is the size of the training data-set and H is the number of experiments to determine the hyperparameters to be employed.

Energy Sustainable AI: SOTA Approaches and Applications

Driven by a common sense for a greener AI, different works have pursued this direction. As an example, a framework meant to track the impact of ML is introduced in [115]: the authors propose a solution combining different metrics, such as training and inference time, to estimate the amount of carbon emissions in Reinforcement Learning. They used specific tools provided by hardware vendors to monitor CPU and GPU power draw, in particular they employed Intel's

RAPL tool with the powercap interface for CPU energy monitoring and Nvidia's nvidia-smi for GPU power consumption estimation. These tools provide real-time energy consumption feedback for a limited range of supported CPU models. In [286] the authors address the problem of an energy-efficient resource allocation in the FL process, formulating it as an optimization problem where the goal is to minimize the total energy consumption of the system under latency constraints. The solution is assessed with 50 users, uniformly distributed in a square area of size 500mx500m with a base station (BS) located at its centre. They proved the consistency of the optimized algorithm comparing it with canonical FL. The work done demonstrated the higher efficiency of the proposed solution in terms of completion time and energy consumption.

A hierarchical EFL solution is proposed in [161] where the authors illustrate an iterative and partial, weighted aggregation process used to speed up the ensembling phase and to reduce energy costs. They show that their HierFedAvg, an extension to the classic FedAvg algorithm, achieves faster convergence on both IID and non-IID samples. They perform their experiments on the MNIST and CIFAR-10 datasets, showing that HierFedAvg simultaneously reduces the model training time and the energy consumption of the end-devices.

Early endeavors in frugal learning focused on compressing the training set, represented as a decision table, onto resource-constrained devices. This approach, currently flourishing in the realm of binary decision diagrams pioneered by Minato et al. [186], involves mapping the training set into structures like Field Programmable Gate Arrays (FPGAs) [150]. Another remotely related line of work explores approximate knowledge bases [83], addressing scenarios with large or complex knowledge bases (KB). This method aims to construct lower (KBL) and upper (KBU) bounded KBs, creating approximations for KB decisions based on whether KBL(x) is true, KBU(x) is false, or the result is unknown. Another field within Machine Learning connected to frugality is online learning, where examples are processed one at a time [103]. The critical challenge is associated with potential changes in the sample distribution and the subsequent adjustment of the hypothesis. The trade-off revolves around the number of examples employed by the learner: detecting distribution changes early (discarding outdated examples) while maintaining robustness against example noise. Additionally, certain online algorithms for high-speed data streams opt to use only a subsample of available examples to expedite training [73].

Some approaches in Machine Learning aiming for frugal decision-making adopt a two-step process. Initially, a potentially computationally expensive model is learned, followed by the simplification of this model in various ways while retaining its predictive accuracy as much as possible. In [107], for instance, features associated with a linear classifier having low absolute weight values are considered poorly relevant or redundant, leading to their removal, potentially through an iterative process. Busa-Fekete et al. [31] focus on the extensive set of hypotheses acquired during a boosting process. Traditionally, these hypotheses contribute to the final decision through voting. However, it is possible to selectively prune the set of hypotheses in an example-dependent manner, transforming the ensemble into a Directed Acyclic Graph. An illustrative example is provided by Urban et al. [257], focusing on compressing deep neural networks [32]. In this scenario, a substantial ensemble of large and deep neural networks, known as the "teacher" is trained from a specific dataset E. This ensemble is employed to label a much larger set of samples E. In their work, Urban et al. [257] demonstrate that by utilizing this new training set, which consists of samples from E along with their corresponding teacher labels, it is possible to train a shallow neural network with equivalent performance as the teacher, achieving significant reductions in the number of weights, although the process may not be strictly considered frugal. This process is currently know as Knowledge Distillation [117].

In the subsequent sections, we originally propose two different frameworks shedding light on the integration of AI technologies in energy constrained environments comparing achieved results against the current state of the art.

3.2 Towards Energy-Aware Federated Learning: Adaptive Round Planning for A Distributed Learning

Taking the initial stride towards an energy-conscious AI process, we take Federated Learning (FL) as a state of the art use case and we introduce an innovative distributed framework designed to collect real-time and detailed process metrics for intelligent process management. Within this context, we explore pertinent FL round planning strategies, which are categorized as static (involving a fixed number of iterations) and dynamic (determined at runtime based on trade-offs between model accuracy and energy expenditure). To substantiate our approach, we outline the blueprint of an experimental testbed that demonstrates an Energy-Aware Federated Learning (EFL) process implemented on actual data. The testbed highlights the capabilities of our solution in terms of metric collection and subsequent adaptive planning. Filling the existing gap, we present the design of a novel distributed framework capable of collecting accurate (worker) energy expenditure and learning-centric metrics at each FL round. The framework comprises state-of-the-art technological building blocks, purposely integrated to enable advanced and energy-aware FL process orchestration capabilities. To validate the approach, we rely on a heterogeneous experimental testbed, and conduct a distributed learning process employing a realistic dataset. The preliminary evaluation results reported in this paper highlight the potential advantage in terms of overall energy consumption reduction and the suitability of an adaptive learning framework capable of autonomous evaluations of the most appropriate trade-off to apply between accuracy and energy expenditure.

Framework architecture

To the purpose of providing an innovative FL framework with energy awareness as described in the previous section, we describe the main components of our system:

- The AI Engine, a core module of the framework that provides support for the main ML libraries and allows the development of use case specific AI models. It handles the distributed weight update phase while keeping the overall overhead as low as possible by taking into consideration the energy consumption measured by participating nodes.
- The monitoring module provides a complete and real-time state update from workers in terms of current energy consumption and accuracy score achieved.
- The planner (orchestrator) is the entity in change of actuating the rules designed for FL round management, the control logic of the process.

The design of our solution allows each module to be easily extendable and integrated within existing frameworks and libraries.

The AI Engine handles the coordination of workers over the learning rounds through RPC interaction. The action-oriented nature of this architectural style coincides with purpose of EFL of reducing as much as possible the amount of information shared among the learners. This choice is also confirmed by similarity with some other widespread FL frameworks, which have followed the same direction. PySyft [4], Tensorflow Federated [166] and Flower [132] are some relevant examples of RPC adoptions. To this end, we have decided to integrate Flower, an open-source, easy to extend library, and with very limited demand of runtime resources to support its execution. Flower implements the main aggregation strategies for the ensembling and provide easy-to-use APIs for worker communication with the central server. It wraps the main ML libraries like Tensorflow, Pytorch and Scikit-learn while allowing a transparent embedding of customized ML models developed from scratch. In order to complete the learning aggregation, Flower requires a static minimum amount of workers and a fixed number of rounds. The server waits until all the workers are available and triggers the local learning for each of them. The model update is synchronous, and the main aggregation algorithm is FedAverage, a tensor version of weights averaging. A few well-known vendors have instead decided to move towards lazier connection between server and clients, by adopting a REST approach. For instance, NVIDIA Clara Imaging is REST-based and is a computational platform that eases the development and the deployment of intelligent medical imaging workflows. About RPC, all the frameworks listed above integrate a variant of the canonical RPC, called Google Remote Procedure Call (gRPC). gRPC is a framework for implementing RPC APIs via HTTP that aims to reduce communication overhead thanks to the parsing of descriptive manifest polyglot (i.e., implemented with different technologies) scenarios. In fact, gRPC employs protocol buffers rather than JSON XML manifests as the interface language for serialization and transmission. By doing so, the resulting protocol is more lightweight and flexible if compared with the canonical RPC transmission pipeline.

The monitoring module enables ML experiment reproducibility and logging. The information is collected by exploiting and integrating a state-of-the-art MLOps tool. MLOps, which stands for Machine Learning Model Operationalization Management, aims to reproduce in the ML area the same advantages driven by DevOps in distributed systems deployments. At the beginning of the ML research area or in only research-oriented deployment scenarios, ML models were only tested and developed in isolated experimental systems: then, when the targeted result was considered achieved (e.g., the training phase was considered sufficient and adequate), the model was deployed in the production environment as a canonical remote service. Nowadays, in many AI-enabled industry scenarios, the old and simple deployment pipeline described above exhibits its non-negligible weaknesses. In fact, once an ML model is deployed, it has to be maintained and updated, otherwise it will lead to a degeneration of accuracy over time (e.g., because of changes in the real-life data that cannot be seen during the model training). In addition, since the same model architecture has to be shared among all the workers, the same framework and same package versions have to be reproducible on each participating node. To this end, we select MLFlow as MLOps framework for supporting the whole ML life-cycle [1]. MLFlow is open source, container friendly, widely used and frequently updated. It consists of four different modules: MLFlow Tracking to record and query experiments, ML Projects to package the code in a reproducible format, MLFlow Models to deploy ML models in diverse serving environments and a Model Registry to store and manage models in a central repository. Furthermore, it provides a remote HTTP server endpoint to log ML parameters and results from distributed and decentralized workers which is exactly what we aim to do in EFL. Our framework integrates the ML metrics with energy monitoring information. Despite several different tools exist, almost none of them keeps track of the real energy draw of the considered hardware components. In fact, most of them base their reports on different metrics like CPU and GPU usage, network bit rate and computation latencies. The only exceptions we found are NVIDIA-smi for real time GPU power consumption and PowerTop for subprocess energy draw estimation [2]. To this end,

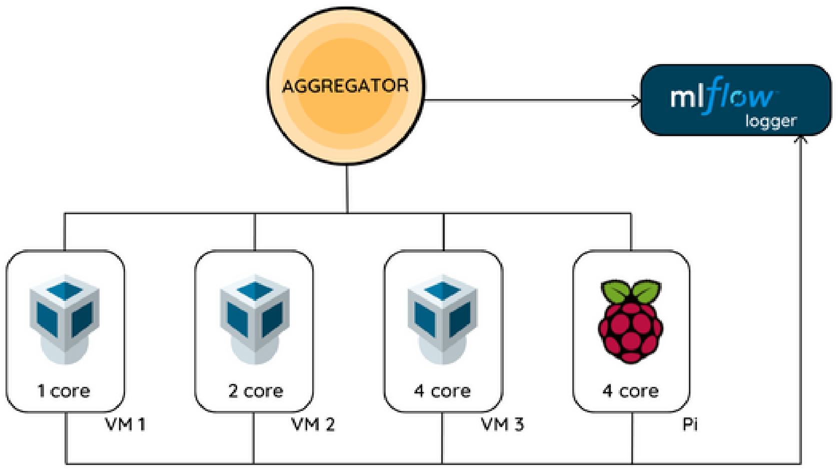


Figure 3.2: Testbed layout and architecture.

we integrated PowerTop as the main source of power consumption information. PowerTop observes power consumption at hardware component level and allows VM energy monitoring, too.

The planner contains all the business logic for the FL dynamic management. Here, the metrics collected by the monitoring module are evaluated over the time in a real-time process which seeks a tradeoff as possible between energy consumption and target accuracy. It autonomously computes an accuracy threshold, which varies over the time by considering a decay factor. The accuracy target thus approaches the actual accuracy with the passing of rounds taking into account the cumulative energy consumption over the B_e budget. The container-based approach allows our framework to be deployed on heterogeneous devices, like those that are widely adopted in edge and IoT scenarios. By delving into finer implementation details, Figure 3.2 illustrates a deployment scenario, depicting all the components involved. An exemplary deployment environment (which will be used for the evaluation testbed described in the following parts of the paper) includes four worker nodes, an ensembler node, and an HTTP logging server for training parameters tracking and results storing. MLFlow logging capability is used to keep track of energy consumption metrics collected from the workers and the ensembler during training and inference phase.

Flexible Support to Differentiated EFL Round Planning Strategies

Herein we discuss potential round planning strategies enabled by our framework, covering also relevant trade-offs one needs to consider. Since the accuracy of the FL model mainly grows with the number of learning rounds - as a coarse-grained approximation, it increases proportionally to the number of rounds - the total number of rounds is a relevant parameter to tune in order to achieve a proper efficiency/accuracy trade-off. It is noteworthy to point out, that none of the surveyed FL frameworks allows a user/developer to dynamically set the number of rounds and workers, that are static parameters, and set to a predetermined value before the beginning of the learning process. As a first distinction, it is useful to differentiate between static and adaptive EFL round planning strategies. In the static EFL approach, the number of workers for each round and the number of rounds are fixed; while in dynamic EFL round planning, instead, these parameters can be tuned according to a desired strategy during the learning phase. Regarding the adoption of the fixed approach, this relies on two basic assumptions: the amount of energy consumed by each node has to be known since the onset of the learning, and it has to be constant during the whole training phase. While the first assumption can be potentially estimated depending on the complexity of the deployment, it is not reasonable to *a priori* assume that energy consumption is constant for each round during the training phase. From this basis, we opt for a more flexible and adaptive approach, taking into account not only energy considerations, but also a level of desired model accuracy.

While preserving the general aspect of our study, without loss of generality, given the starting static parameters: B_e the total energy budget and A_{min} the minimal desired model accuracy; the minimal accuracy target for the next round $A_{min(i)}$ is dynamically computed as:

$$A_{min(i)} = A_{min} \cdot \left(1 - \frac{E_i}{B_e}\right) \tag{3.2}$$

where E_i is the cumulative amount of energy consumed until round i.

In the current implementation, the target accuracy for each round is monotonically reduced by a factor which is proportional to the ratio between the energy consumed until the current round and the total energy budget. Intuitively, the more energy is consumed during a learning round, the lower the resulting target accuracy to reach. It is noteworthy to point out that more advanced process control logic could be envisioned, considering additional decision variables and/or node selection strategies. This study is left as a future work. For completeness, Algorithm 1 sketches the FL process governed by the above control logic. In this setting, the training can be stopped even if the A_{min} is not reached because the $A_{min(i)} \leq A_{min}$ for each round i.

Evaluation and Empirical Results

To assess the validity of our solution, here we present two different results. First, we assess the capability of acquiring up-to-date information regarding power consumption. This is done to

Algorithm 1: Energy Aware algorithm for round planning

```
Input: $A_{min}$, $B_e$
1 $keep\_on\_training = True$ ;
2 $A_{min(i)} = A_{min}$;
3 while $keep\_on\_training == True$ do
4     FL distributed training();
5     Compute $A_i$;
6     Update $E_i$;
7     $A_{min(i)} = A_{min} \cdot (1 - \frac{E_i}{B_e})$;
8     if $A_i >= A_{min}(i)$ or $E_i >= B_e$ then
9         $keep\_on\_training = False$;
```

validate the need for a more flexible and dynamic approach, considering energy consumption as a process decision variable. Second, we adopt the process decision planning driven by Equation 3.2, showcasing the reduction of the accuracy goal to a more energy saving target. For the experimental assessment, we deploy a training process conducted on a real dataset

	MAE	$train_t$	inf_t	**E**
1 core VM	0.0255	11.2 s	8.900e-5 s	203mW
2 cores VM	0.0271	9.3 s	4.600e-5 s	237mW
4 cores VM	0.0305	7.4 s	6.900e-5 s	674mW
4 cores Pi	0.0217	12.5 s	9.730e-4 s	1.3W
Ensembler	0.0322	-	1.300e-5 s	0.3W

Table 3.1: Experimental results with 10 learning rounds

containing solar power plant data presented in [3]. In specific, the dataset includes solar power plant data collected in India over a 34 days period, where each sample corresponds to the amount of energy produced by the electrical inverter connected to a singular solar panel. We preprocess the data, splitting the original one accordingly to the inverter id parameter, in order to have four different datasets fed to each of the workers. This simulates the collection and the processing of local data on the inverter node.

The worker training process adopts a stochastic gradient descent regressor developed with scikit-learn, estimating the amount of energy which will be produced at a future point in time by the local inverter. In order to better mimic a realistic edge case scenario, the testbed is composed of heterogeneous nodes. Specifically, of four worker nodes, three of them are virtual machines,

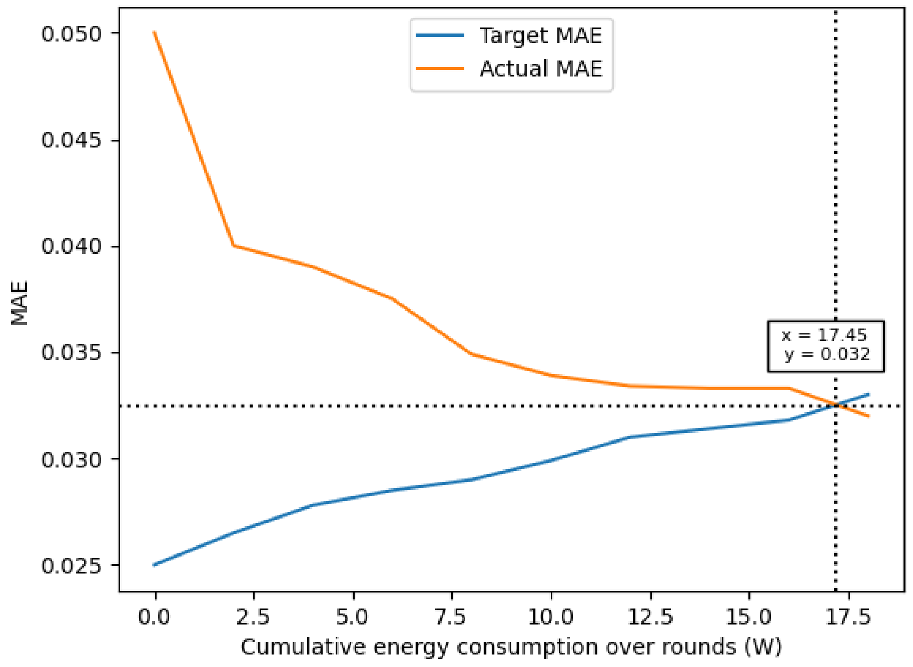


Figure 3.3: Target and actual MAE value in relationship with the required energy consumption.

while the fourth is a real hardware device, a Raspberry model 3. In terms of computational power, VMs are equipped with one, two, and four CPU cores respectively; the Raspberry Pi, instead, is equipped with a Broadcom BCM2837 64bit Quad Core Processor. Regarding the first assessment, Table 3.1 shows the parameters collected by our framework, in particular *MAE* (Mean Absolute Error) as an indicator of the accuracy of the model; E is the amount of energy consumed by the device over the rounds; $train_t$ and inf_t are the training and the inference time, respectively.

Lastly, we checked how our framework adjusts its accuracy goal at each learning round accordingly with Formulae 3.2 while still achieving an acceptable accuracy level. To this end, we consider *MAE* as an accuracy performance indicator to minimize and we summed up the total amount of energy consumed by workers and the ensembler required to reach a certain *MAE* score. For the experiment we set B_e, the overall energy budget, to 20 Watts; and A_{min}, which is here indicated as a maximum *MAE* accepted, to 0.025. In specific, A_{min} and B_e values have been empirically fixed to the *MAE* score and the corresponding energy demand recorded by training the same distributed model architecture on a centralized node.

Figure 3.3 shows the target *MAE* and the actual *MAE* in relationship with the cumulative energy consumption at round *i*. As it can be observed, the two lines intersect at round ten, when the current *MAE* is equal to 0.032, and the cumulative energy consumption over rounds is equal to 17.45 Watts. As a result, the aggregated model finds a sub-optimal solution in a dynamic way by setting a compromise level between the initial accuracy target and a one which is less impactful on the overall energy consumption.

3.3 Serverless FL Orchestration on Heterogeneous and Constrained Devices

Despite the great strides made in terms of precision and privacy awareness, the real adoption of FL in real-world scenarios, particularly in industrial deployment environments, is still an open thread. This is mainly due to the additional complexity that stems from notoriously long and costly development cycles when employing AI techniques on bandwidth-, computing-, and energy-constrained nodes. At the same time, quickly developing edge AI applications that are cloud-native, flexible, and secure has never been more important. Motivated by these challenges, we address scenarios involving highly heterogeneous computing capabilities and energy budgets, presenting EneA-FL, an innovative scheme for serverless smart energy management. This approach dynamically adapts to optimize the training process and facilitates seamless interaction between Internet of Things (IoT) devices and edge nodes. The proposed middleware includes a containerized software module that efficiently manages the interaction of each worker node with the central aggregator. EneA-FL, by monitoring local energy budgets, computational capabilities, and target accuracy, makes informed decisions about the inclusion of specific nodes in subsequent training rounds, effectively balancing the tripartite trade-off between energy consumption, training time, and final accuracy. In extensive experiments across diverse scenarios, our solution demonstrates impressive results, achieving 30% to 60% lower energy consumption compared to popular client selection approaches in the literature, while being up to 3.5 times more efficient than standard FL solutions. Crafted to tackle the energy management challenges inherent in real-world FL deployment, this innovative middleware streamlines the training and deployment of FL models across diverse IoT devices. At its core, an orchestrator dynamically oversees the FL process, dispatching a containerized environment to participant nodes for real-time monitoring of computing and networking capabilities, energy budget, and local accuracy. The orchestrator autonomously evaluates the effort required by each IoT client to meet specified QoS, minimizing the need for user or developer intervention. Notably, EneA-FL introduces a dynamic coordination approach for participating nodes, optimizing energy efficiency. Tested across various clients and energy requirements, including CPU- and GPU-enabled microcontrollers with limited resources, our automated energy management scheme outperforms standard FL approaches.

Modelling Energy Consumption of FL Worker Nodes:

In a broad sense, if we denote $E(i)$ as the energy consumption of a fog node i, and envision the typical FL scenario comprising an arbitrary set of workers with a size of n and a single aggregator node—where, ideally, all working nodes possess identical networking and computational capabilities—we can calculate the total system energy consumption E_s by summing the individual contributions of each worker and the aggregator node. Hence, we can express E_s as:

$$E_s = \sum_i E(i) = n \cdot E_w + E_a,$$

where E_w denotes the energy consumption of a single worker, and E_a represents the energy consumed during the aggregation process.

Upon delving deeper into the contribution of each node in a FL scenario, for a generic worker w, the energy consumption E_w is directly proportional to the local model complexity, denoted as $M_c(w)$, and the size of the local dataset, referred to as $S_d(w)$.

In a FL scenario, where all participating nodes share the same model architecture, the local model complexity is uniform across workers:

$$M_c(w_1) = M_c(w_2) = \ldots = M_c(w_{n-1}) = M_c(w_n) = M_c.$$

On the server side, for the aggregator node a, the computation E_a typically involves an average-like operation, with a limited computational cost. However, this operation can become expensive with a high number of tensors to be averaged. Therefore, E_a is directly proportional to the number of workers, n.

The energy modeling scheme described above would be incomplete without an explicit acknowledgment of the relationship between energy, latency, and accuracy. While the primary goal of a FL pipeline is to achieve the highest possible accuracy while minimizing overall latency, optimizing these constraints becomes more challenging in the presence of heterogeneous working nodes.

In an ideal scenario with low heterogeneity among participating nodes, the optimization objectives would be straightforward:

$$\begin{cases} \min(\Delta Lat) \approx \min(M_c) \\ \max(Acc) \approx \max(M_c) \\ \min(E_s) \approx \min(M_c) \end{cases} . \tag{3.3}$$

In this case, minimizing M_c would benefit E_w, E_a, and ΔLat. The only trade-off to consider would be between Acc and M_c to minimize energy consumption while satisfying Quality of Service (QoS) requirements.

However, the dynamic and erratic nature of the fog environment introduces a higher-dimensional space of possible solutions. As each worker node may be involved in multiple tasks and may have a dynamically changing percentage of bandwidth at its disposal over time, selecting a specific node during the aggregation phase becomes challenging and can have negative impacts not only on Acc and ΔLat but also on E_s. For this reason, selecting nodes dynamically over training rounds becomes fundamental.

EneA-FL Architecture:

EneA-FL is the pioneering middleware that effectively combines the strengths of serverless and fog computing, demonstrating its suitability for Federated Learning (FL) scenarios characterized by the presence of highly dynamic and heterogeneous devices. EneA-FL comprises three core modules (see Fig. 3.4 as a reference):

- **Energon**, built upon the well-established open-source toolkit Prometheus[1], serves as a Prometheus-compliant exporter designed for IoT and edge devices. It functions as a container shipped to participating nodes, transparently collecting energy metrics independently of the local operating system. Energon provides an endpoint for aggregators to poll energy and network metrics, scraping data from various Linux-based microcontrollers. Published as a Pypi package[2], Energon enhances community accessibility to edge device system metrics;

- **Furcifer** is an innovative container orchestrator facilitating communication among participating nodes through an overlay network. Leveraging Docker DNS interface, it

[1]https://github.com/prometheus/prometheus
[2]https://pypi.org/project/energon-prometheus-exporter/

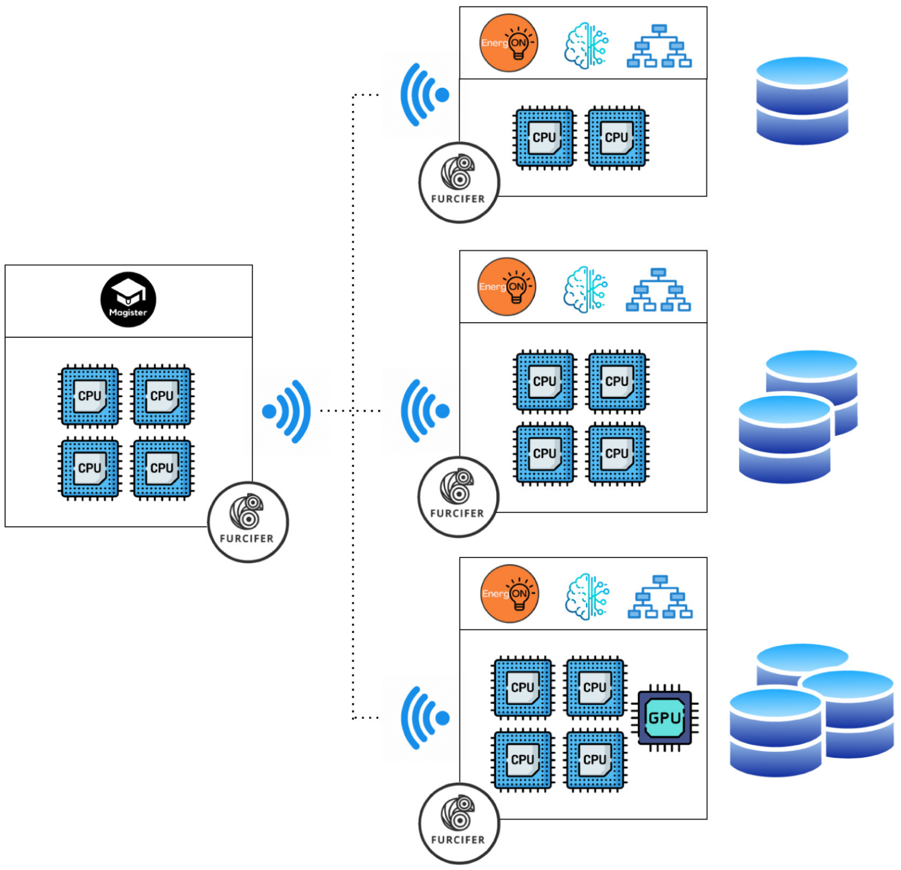


Figure 3.4: EneA-FL, serverless middleware architecture

establishes peer-to-peer communication between nodes and their corresponding containers within the cluster. Furcifer additionally equips each participating node with a kernel-compliant, container-based application. Additional details about this module are provided in Section 5.2;

- **Magister** serves as the policy manager, overseeing the aggregation process. It is responsible for selecting aggregation policies, determining participant clients for each aggregation round, and deciding when the learning process meets specified Quality of Service (QoS) requirements. Magister plays a crucial role in orchestrating the completion of the learning process based on the satisfaction level of QoS requirements for the FL application.

Energon: Transparent Energy Monitoring

Energon is a modular monitoring tool for IoT and edge devices. It keeps track of an extensible set of system metrics about energy consumption, network channel quality, and resource utilisation, among others. Collected metrics are compliant with the Prometheus exporting standard, which was recognised as graduated project maturity level in 2016 by Cloud Native Computing Foundation, the open-source vendor-neutral hub of cloud-native computing. Diagnostic information can be obtained by HTTP requests to the */metrics* endpoint on the IoT device. This allows both scanning with application-dependent business logic and a smooth interaction with the Prometheus ecosystem.

The ubiquitous nature of Energon allows the user to monitor critical metrics and to make real-time decisions at the application level, without generating any additional overhead for the constrained devices. When it comes to system monitoring, one of the primary objectives is to minimise the additional operations necessary to collect the desired metrics while the target applications are running. Energon has been meticulously designed to ensure complete isolation from the rest of the system. It operates independently and does not require any interaction with the running applications, thereby eliminating any potential interference or performance overhead caused by monitoring processes. Running as a separate process allows Energon to efficiently collect the desired metrics and perform monitoring operations without being tightly coupled to the application's execution and its business logic. By adopting this approach, Energon efficiently and seamlessly gathers essential metrics without impacting the performance and behaviour of the monitored applications, making it an effective and non-intrusive solution for system monitoring tasks. This helps developers focus solely on the development of their applications without the need to worry about logging the device's state for later historical analysis or real-time decision-making.

In addition to raw data monitoring and exposition, Energon can be customised to send an event when a specific condition is met. Inside our EneA-FL middleware, Energon plays the central role of keeping the orchestrator updated about the current state of the monitored nodes, thus allowing the policy manager (i.e., Magister) to select the best workers available in terms of residual energy, instantaneous power consumption, and peer-to-peer communication quality. About the querying interface, Energon wraps PromQL, the functional expression language defined by Prometheus, with easier high-level REST APIs. Those JSON-based endpoints can be further customised depending on QoS requirements. Scraped metrics can be stored locally on the orchestrator side, as done inside EneA-FL, or they can be saved on a separate database for later use—e.g., time series analysis for designing new better policies. All metrics are stored as time series data identified by a metric name and a set of key-value pairs. Sharding and federation are also possible with minimal additional settings.

Magister: Context Aware Decision Making for the Edge

Within EneA-FL, Magister operates as a pivotal component of the container orchestrator, responsible for optimizing the client selection policy based on individual worker states in terms of system metrics and QoS satisfaction. Magister considers factors such as the energy consumption of participating workers, the time required for local training procedures, and the accuracy improvement compared to previous training epochs. At this stage, communication aspects are not factored into the considerations.

At the onset of each federation round, Magister utilizes Furcifer to gather clients resource usage information from Energon and subsequently selects clients based on the collected metrics. The local training procedure remains unaffected by Magister, and upon receiving local updates from the selected clients, Magister makes decisions on whether to continue or halt the distributed training process, contingent on achieved QoS metrics. Consequently, our solution ensures that the global model aggregation process remains unaffected. Magister thus serves as a flexible client selection component that can seamlessly integrate with any custom training and aggregation mechanism for Federated Learning (FL). To delineate the efficacy of Magister's intelligent selection process, we analyze its impact on consumed energy, execution time, and accuracy improvements in the worker selection process.

The novel worker selection approach introduced by EneA-FL involves optimizing the choice of workers for each round of the federated learning (FL) process. The traditional selection function, denoted as $\mathscr{S}\{\mathscr{W}\}$, is enhanced by Magister to smartly choose impactful nodes, minimizing energy and time consumption while maintaining model performance. The approach considers a history-dependent selection function $\mathscr{S}$ to intelligently select workers based on their past contributions and reliability. To find the optimal selection function, the relationship between the aggregated model's performance and the energy/time spent is explored. The effectiveness metric $\mathscr{E}$ combines energy, time, and performance metrics, allowing for the identification of impactful workers.

In our pursuit of minimizing resource expenditure at the worker level while avoiding a shift in computational burden to the aggregator node, we adopt a simplified approach that obviates the need for solving the combinatorial optimization problem. To achieve this, we define an effectiveness measure designed to assess the impact of the local model update from a worker w_i on the performance and resource consumption of the entire federation setup. The effectiveness metric is formulated as follows:

$$\mathscr{E} = \alpha \cdot \frac{E\left(\mathscr{N}_i^{(t)}\right)}{\max\{E^{(t)}\}} + (1-\alpha) \cdot \frac{\tau\left(\mathscr{N}_i^{(t)}\right)}{\max\{\tau^{(t)}\}} - \beta \cdot \Delta\left(\mathscr{P}\left(\mathscr{N}_a^{(t)}\right), \mathscr{P}\left(\mathscr{N}_{a\ominus i}^{(t)}\right)\right), \tag{3.4}$$

Here, $\max\{E^{(t)}\}$ represents the maximum energy spent by any worker at step t, i.e., $\max\{E^{(t)}\} = \max\{E\left(\mathscr{N}_j^{(t)}\right) \forall j\}$. Similarly, $\max\{\tau^{(t)}\}$ signifies the maximum time taken by any worker at step t, denoted as $\max\{\tau^{(t)}\} = \max\{\tau\left(\mathscr{N}_j^{(t)}\right) \forall j\}$. The term Δ represents the performance difference between the aggregated model $\mathscr{N}_a^{(t)}$ and the model $\mathscr{N}_{a\ominus i}^{(t)}$ obtained by aggregating updates from all workers except w_i. Finally, α and β are hyperparameters that govern the trade-off between the significance of energy consumption, time requirements, and the achieved performance improvement.

Testbed and Experiment Preliminaries

The experiments are executed on a practical networking testbed featuring a diverse array of heterogeneous edge devices, effectively emulating a real-world deployment environment with varying computational capabilities. We leverage representative datasets from LEAF [44], encompassing diverse use cases in Computer Vision and Natural Language Processing domains. Specifically, we choose two datasets, MNIST and Sent140, from LEAF to assess the viability of our energy-aware Federated Learning selection across multiple tasks. MNIST, a distributed version of the well-known MNIST dataset, is employed for image classification. In addition, Sent140 comprises a corpus of 1,600,000 tweets obtained using the Twitter API for sentiment analysis. Lastly, we incorporate the N-BaIoT dataset [178], which contains realistic traffic data collected from 9 commercial IoT devices authentically infected by Mirai and BASHLITE.

Then, perform a comparison between the containerised FL training application and the execution at operating system level. This reveals notable similarities in their energy consumption, with a slight difference of approximately 5% in terms of training time. As illustrated in Fig. 3.5, both executions exhibit comparable energy usage, suggesting that the containerization process does not have a significant impact on overall power consumption during training tasks. Containers, serving as self-contained entities encompassing application code and dependencies, wield substantial influence on the responsiveness of IoT applications during initialization and deployment. To gauge and compare startup latencies with and without GPU hardware acceleration, we delve into the potential performance enhancements and resource optimization within serverless architectures for IoT applications. This section meticulously examines the minimal additional overhead introduced by containerization during training execution, with a focus on

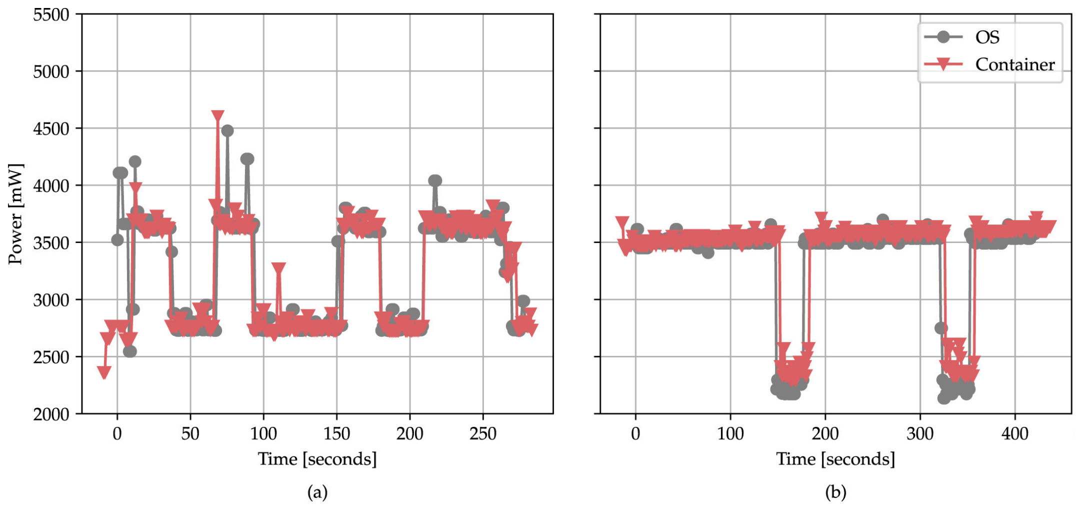


Figure 3.5: Comparison of power consumption of containerized workers vs OS level execution. (a) On Sent140 dataset. (b) On N-BaIoT dataset.

assessing startup latencies across a spectrum of devices through over 100 startup cycles. We first evaluate the energy consumption of a wide range of IoT devices commonly used in fog and edge scenarios. In particular, we measure the energy consumption of our models on a Raspberry Pi model 4, a Jetson Nano developer kit, a Jetson Xavier NX board, and a Jetson AGX Orin developer kit. In addition, each of the Jetson boards has been tested with and without GPU support to check the energy cost of hardware acceleration in IoT training processes. Startup latencies across variously configured devices demonstrating that the incorporation of hardware acceleration does not yield significantly higher delays. On average, the supplementary latency attributed to hardware acceleration stands at a mere 5%–10%. Furthermore, across all devices assessed, latencies consistently fall within the range of 200 ms to 300 ms, emphasizing the negligible impact of containerization techniques on the overall efficiency of training execution.

Figure 3.6 visualizes the experiments conducted to gauge the energy consumption of all devices across five training epochs. The Jetson AGX Orin developer kit emerges as the swiftest, albeit exhibiting higher instantaneous power consumption. Conversely, the Jetson Nano without GPU stands out as the slowest device, whereas the hardware-accelerated iteration of the device proves to be the optimal compromise in balancing energy consumption and training time.

Effectiveness Parameter Space Exploration and Corresponding Results

To assess the impact of the hyperparameters α and β on the federation optimization, we conduct experiments by varying their values between 0 and 1, and 0 and 100, respectively. We perform 10 federation optimization experiments for each combination of α and β, setting the number

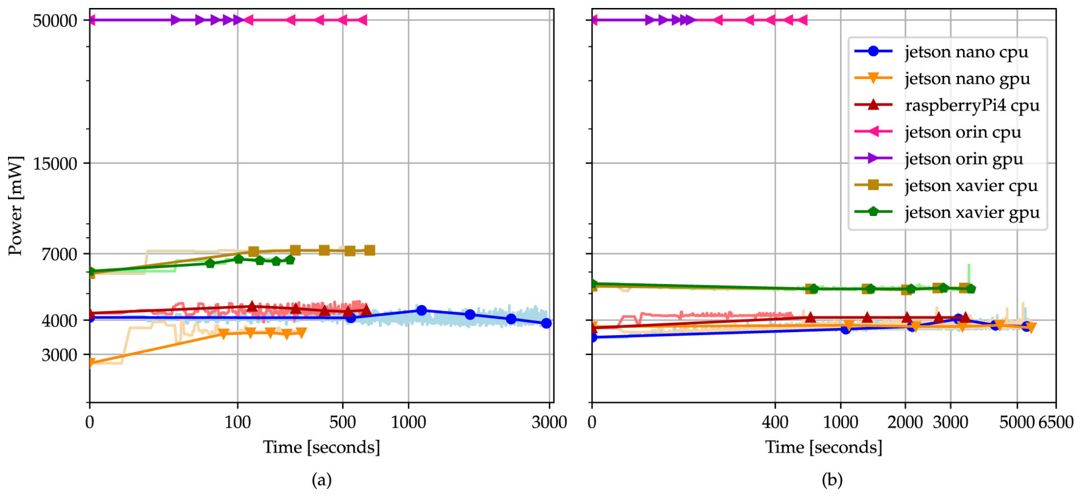


Figure 3.6: FL workers power consumption over tested devices. (a) On Sent140 dataset. (b) On N-BaIoT dataset.

of federation rounds to 30. For each experiment, we record (i) overall energy consumption, (ii) total execution time, and (iii) the number of rounds required to achieve convergence. Convergence is defined as reaching 97% accuracy on the test set for the global model. In our experiments, we identify that setting α to 0.6 and β to 40 represents the optimal configuration for EneA-FL. This setup enables the system to consume only 1.3 MJ of energy, complete 30 federation rounds in 12.7 hours, and achieve convergence in 10.8 steps. Having obtained the best α and β setup, we compare our proposed solution with the standard FL setup – which relies on random node selection – and the OORT [149] and its extended version (OORTv2) [16] in 3.2. The findings emphasize that Magister provides a favorable solution for reducing both energy consumption and execution time when compared to all baselines. Importantly, Magister achieves these improvements without affecting convergence time, measured as the number of rounds required to reach 97% accuracy. In addition the EneA-FL policy vastly outperforms the selected baselines, achieving higher accuracy over all datasets and showing a statistically significant higher longevity in terms of residual energy consumption. This is confirmed by observing

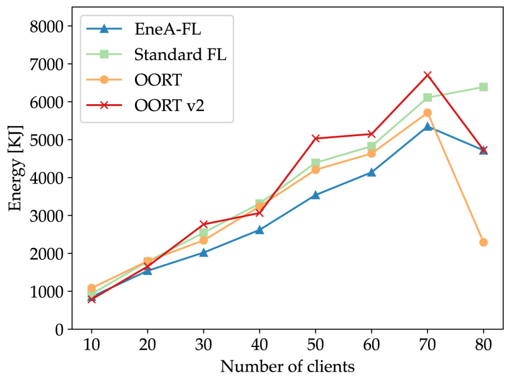
Table 3.2: Comparison with other state of the art solutions.

Approach	Energy	Time	Rounds
Standard FL	4.6 MJ	28.5 h	9.7
OORT [149]	1.8 MJ	13.2 h	11.2
OORTv2 [16]	2.2 MJ	13.8 h	13.8
EneA-FL worst	2.1 MJ	14.1 h	10.7
EneA-FL best	1.3 MJ	12.7 h	10.8

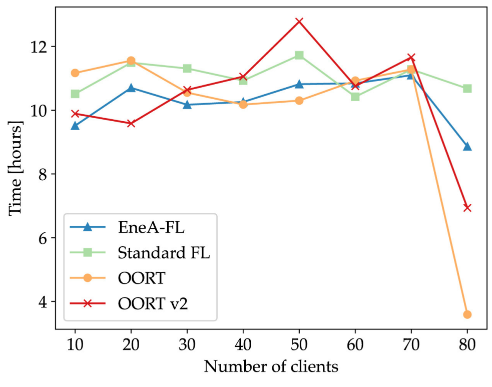
Fig. 3.8, where we depict the average distribution of selected devices over federation rounds during MNIST training. On the x-axis, it is apparent that the baselines last for a maximum of 8 federation rounds, whereas EneA-FL extends to as many as 17 optimization rounds.

Then, we investigate the influence of the number of clients per round on the federation. We explore the selection of n workers per round, allowing n to vary between 10 and 80. In total, the federation comprises 100 devices, so n spans from choosing only a small subset of federation workers to nearly selecting all of them. In more detail, the results related to energy consumption are presented in 3.7. It is noteworthy that consistently outperforms all selection baselines for nearly every value of n. For smaller n values, such as $n = 30$ and $n = 40$, EneA-FL requires almost 30% less energy to achieve convergence to the same accuracy level. However, as n increases, including more workers in the selection, Magister ends up choosing both efficient and inefficient devices—given the limited number of efficient devices—thereby diminishing the advantage of smart device selection.

Finally, we study the percentage of dead selected devices over the federation round, comparing EneA-FL with the selected baselines. We model the discharging process of devices as an exponentially distributed event over the federation rounds that a device can complete and set its average value to be equal to a random value between 1 and 5. Therefore, in this experimental setup, each device belonging to the federation is capable of completing a variable number of federation rounds, after which it discharges completely and stops sending updates if selected. As depicted in Figure 3.9, the number of selected devices that cannot produce updates initially increases in the first few steps of the federation process for both approaches. In particular, the baselines continue selecting devices unable to produce updates, reaching peaks of up to 80% of selected workers with drained batteries. In contrast, EneA-FL employs its energy management approach, resulting in a stable percentage of selected workers with dead batteries, always staying below 40%. This is achieved through the handicap given to devices that do not produce any updates. Additionally, the energy management mechanism contributes to a more rapid convergence of the federation, as updates are consistently produced by workers.



(a)



(b)

Figure 3.7: Impact of the number of selected clients per round on energy consumption (a) and execution time (b) for the MNIST dataset.

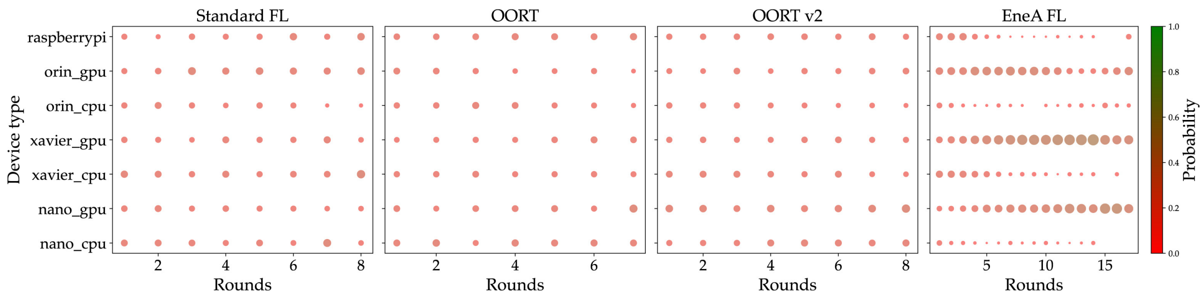


Figure 3.8: Worker lifetime comparison again other experimental contributions.

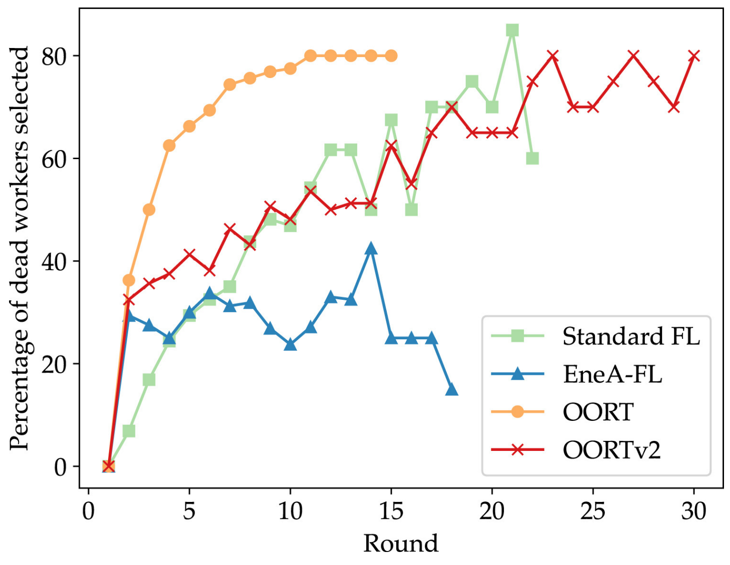


Figure 3.9: Worker discharge percentage analysis compared with baselines.

Chapter 4

From Big Data to Small Data: Practicing Frugality in Constrained Data Environments

It is very reasonable to think that we are going to witness an increasingly higher amount of data collected to feed more complex predictive models. As a consequence, the current way employed to collect and process data is going to be unfeasible to handle the incoming amount of information transferred over the network. Despite the current enthusiasm for Big Data, some studies have been discussing new paradigms based on learning partial differential equations from "Small Data" instead of training over huge data-sets. As the study done by Maziar Raissi et al. [210] has shown, it is possible to design learning machines with the ability to operate in complex domains without requiring large quantities of data. From that perspective, a Small Data approach satisfies both the requirements in terms of privacy awareness and incremental improvement of ML models, segmenting the data distribution and grouping data-sets into clusters with similar characteristics or profiles. In this sense, two different approaches can be evaluated in order to increase the capabilities of a predictive model:

- **Arbiter approach**: a single entity has to weight the contribution of each participating node basing its evaluation on previous prediction results.
- **Combiner approach**: a combiner can be seen as an entity in charge of learning and applying a composition rule. This strategy aims to coalesce the predictions from the participating node models by learning the relationship or correlation between these predictions and the correct one. A combiner computes a prediction that may be entirely different from any proposed by a single model, whereas an arbiter chooses one of the ones already computed by the participating nodes.

While the former comes with various limitations such as single point of failure and questionable trust level of participants, the latter lacks a standard way to combine smaller models into a comprehensive resulting ensemble.

In Section 4.1 we analyze the main challenges of AI applications in data constrained environments while exploring the current state of the art. In addition, this section encompasses two experimental contributions compared with the state of the art: in Section 4.2, we introduce a blockchain-enhanced FL framework aligned with the aforementioned *Arbiter approach*; and in Section 4.3 we employ an ensemble of weak autoregressive algorithms for predicting traffic patterns in the context of smart cities, more oriented towards the *Combiner approach* we just discussed.

4.1 Challenges in Deploying AI in Constrained Data Environments

In this section, we delve into *Small Data* existing methodologies as a substitute for traditional approaches focusing on privacy concerns that arise when data is scarce and dispersed among participants. Then, we explore how, in such scenarios, algorithms with lower complexity can outperform deeper neural networks.

4.1.1 Trust in Distributed Learning

Traditional machine learning (ML) approaches, relying on centralized data processing, face impracticality when dealing with diverse and substantial data from various locations. This challenge is notably highlighted by privacy concerns [191], and legal regulations, exemplified by the European General Data Protection Regulation (GDPR) [81], create obstacles for centralized ML methods due to the potential risk of data leaks. In response to these challenges, Federated Learning (FL) emerges as a valuable solution. Unlike centralized ML, which typically relies on cloud-based resources and involves sending data to a central entity, FL conducts training directly on remote clients using their on-premises data. Each client independently trains a local ML model with its data and subsequently transmits it to a server. The server aggregates all received partial models according to a predefined strategy [25].

While FL addresses certain AI challenges, such as preserving privacy, it is not immune to attacks, including model poisoning and inference attacks [245]. Trustworthiness remains a significant concern, especially with the potential presence of malicious clients and servers that can negatively affect FL training and introduce backdoors [153]. Moreover, the traditional FL architecture based on the client-server model suffers from a single point of failure, low scalability, and susceptibility to tampering of the global model, leading to possible biases in favor of some partial models over others [194]. In FL processes, ensuring transparency is crucial for clients, particularly when there is a lack of complete trust among participants or with the service provider. Concerns may arise regarding the node responsible for aggregating models, which could introduce biases and favor certain updates over others. Additionally, there is a need to verify the integrity of models submitted by clients, as malicious contributions could compromise the overall performance of the global model or introduce backdoors [19]. Therefore, to facilitate collaboration among unknown clients effectively, service providers must guarantee two key aspects: (i) fair treatment of all partial models without biases that could favor specific contributions, and (ii) robust mitigation measures against malicious attempts to manipulate the global model through a rigorous validation process for partial models.

Addressing Trust in Distributed Learning: SOTA Approaches and Applications

In the realm of privacy-preserving techniques for Federated Learning (FL), one prominent approach is Differential Privacy (DP), as highlighted in the literature [271]. DP involves introducing artificial noise to the partial model before transmitting it for aggregation. Striking a balance between preserving privacy, as indicated by the level of noise introduced, and maintaining satisfactory global model performance becomes crucial. Adjusting this tradeoff is essential to ensure adequate privacy guarantees without compromising the overall efficiency and convergence time of the training process. Another emerging avenue in FL focuses on leveraging homomorphic encryption [199]. This encryption technique allows operations to be conducted on encrypted data directly, eliminating the need for decryption before processing. Clients encrypt their partial models before transmission, and the aggregation produces an encrypted global model. Once decrypted, this model serves as a privacy-preserving solution that clients can utilize without exposing sensitive information during the collaborative learning process.

To enhance trustworthiness among participants in Federated Learning (FL), one approach involves selecting clients based on their reputations and assigning weights to partial models according to a trust score associated with each participant. Kang et al. [138] propose a method that evaluates reputation stored on a consortium blockchain during the selection of FL training participants. Reputation, in this context, is measured from the training task completion history, considering past behaviors such as reliable or unreliable activities. While this approach is tailored for mobile devices, it may face limitations in scenarios with a restricted number of clients where discarding nodes in advance is not feasible. In such cases, all contributions, even from a small number of participants, can be valuable for improving the quality of the global model. Cao et al. [45] introduce FLTrust, a Byzantine-robust FL method designed to safeguard against malicious attacks. FLTrust achieves this by training a server model using a small, manually collected clean training dataset as if it were a client. Trust scores are assigned to each local model update based on its similarity with the server model update, and these scores are utilized for weighting local model updates in the generation of the global model. Despite its robustness against attacks, FLTrust has limitations related to the centralized server and its dependency on the training dataset provided to the server. Additionally, there is a risk of assigning low trust scores to honest clients that perform exceptionally well.

Other contributions already introduced the symbiotic interplay between FL and blockchain as a power way to guarantee accountability and fairness. Lo et al. [164] leverage blockchain technology to introduce accountability and enhance fairness in FL systems. By utilizing blockchain, they ensure data-model provenance by storing hashed values of data, local model versions, and global model versions. To address fairness concerns, the authors propose an algorithm that dynamically samples training data from classes with poor representation, guided by the inverse of the weight distribution observed in the dataset used for testing. Chen et al. [53] introduce a decentralized FL framework based on blockchain technology, incorporating a decentralized validation mechanism for partial models. In each round of FL training, a subset of devices is chosen to serve as validators. These validators independently validate all local updates using their respective local datasets. After examining the experimental results, each validator casts votes to assess the legitimacy of each model. Aggregating votes from multiple validators allows for the identification and removal of malicious devices associated with negative model contributions. This approach enhances the robustness of the FL framework, as the validation operations can be effectively conducted even in the presence of compromised validators. Specifically in industrial IoT scenarios, Basset et al. [6] present Fed-Trust, a blockchain-orchestrated edge intelligence framework for trustworthy cyberattack detection in IIoT. However, their approach does not bring remarkable novelties in terms of validation of partial models since the verification phase consists in allowing fog nodes to collect the block comprising the partial models from all contributors to calculate the global model. In this work, trustworthiness is intended as one of the main targets of cyberattacks for the IIoT that can be protected through a distributed temporal convolution generative network.

TrustFed [213] is a fully decentralized cross-device FL framework based on blockchain technology. It ensures fairness by utilizing statistical outlier detection techniques to identify and remove malicious participants from the training distribution. TrustFed incorporates blockchain and smart contracts to manage the reputations of participating devices. However, it's important to note that TrustFed's approach may eliminate outliers, including contributions from clients that have achieved significantly better performance due to having larger local training sets than other participants. Gao et al. [93] introduced SVeriFL, a novel protocol that leverages Boneh-Lynn-Shacham (BLS) and multi-party security for verifying the integrity of partial models contributed by clients and ensuring the correctness of their aggregation. However, similar to previous approaches, the primary focus is on verifying the integrity rather than emphasizing the quality of the submitted partial models. Additionally, the protocol relies on a trusted authority, introducing an element of centralization and potential vulnerability. Li et al. [156] propose a dynamic verification strategy to mitigate the influence of abnormal clients on the global model. Similar to our work, the authors utilize a secondary server-side dataset to validate the contributions of each client, only incorporating partial models that achieve satisfactory accuracy in the aggregation process. However, their approach retains vulnerabilities associated with using a centralized server for model aggregation. Despite recent advancements in techniques ensuring the correctness of partial model aggregation without relying on blockchain and smart contracts, these new approaches are still susceptible to a single point of failure that could compromise the entire system.

4.1.2 Low Complexity Algorithms in Time Series Forecasting for Real-World Scenarios

In the realm of statistics and machine learning, the bias-variance tradeoff characterizes the inherent property of predictive models. It reveals that models with lower bias in parameter estimation tend to exhibit higher variance of parameter estimates across diverse samples, and conversely, models with higher bias often entail lower variance. This tradeoff encapsulates the delicate balance and dilemma faced by practitioners—known as the bias-variance problem—in attempting to simultaneously minimize these two types of errors. This challenge arises as a hindrance preventing supervised learning algorithms from effectively generalizing beyond their initial training set.

Bias, as a component of the bias-variance tradeoff, represents the error stemming from erroneous assumptions within the learning algorithm. When bias is high, indicating overly simplistic models, the algorithm may inadvertently overlook crucial relationships between features and target outputs, resulting in a phenomenon known as underfitting [261]. On the other side of the spectrum, variance, another facet of this tradeoff, manifests itself as an error due to the algorithm's sensitivity to minor fluctuations in the training set [112]. Elevated variance, indicative of overly complex models, can cause the algorithm to excessively capture the random noise present in the training data, deviating from the intended patterns and leading to overfitting.

There are four different reasons why it is then preferable to employ a simpler model over a more complex one:

- **Prevents Overfitting:** Utilizing a simple model is advantageous in preventing overfitting, particularly in high-dimensional datasets with an abundance of features. Overfitting occurs when a model captures not only the genuine patterns in the data but also random fluctuations, leading to reduced generalization performance.

- **Interpretability:** Simple models, with fewer features, enhance interpretability. In the context of an overly complex model with numerous features, especially when these features are correlated, understanding the model's behavior becomes challenging. Simplicity helps to obtain a clearer and more accessible interpretation.

- **Computational Efficiency:** Training a model in a lower-dimensional data set contributes to computational efficiency. The execution of algorithms on simpler models demands less computational time, offering practical advantages in scenarios where efficiency is crucial.

- **Higher Accessibility:** Simple models are often more accessible to a broader audience. Their straightforward structure and reduced complexity make them easier for stakeholders, including non-experts, to comprehend and utilize. In addition, lower computing capabilities are required to train simple models, making the cost associated with computational resources accessible to smaller players.

In this section, we delve into the potential advantages of employing low-complexity algorithms over more intricate ones, such as deep neural networks. Our focus is directed towards scenarios characterized by a limited dataset size, where the risk of overfitting a complex model becomes more prominent.

Data Size and Model Complexity Tradeoff: SOTA Approaches and Applications

Choosing the most suitable model from a set of competing candidates is a common challenge in real-world modeling. The definition of the "best" model is contingent upon its effectiveness in serving a specific purpose related to a well defined dataset. Evaluation often involves comparing the model's simulations to observed data. Model selection methods are employed to identify a balance between achieving a good fit with the data and maintaining an appropriate level of model complexity. The diverse interpretations of model complexity inherent in various model selection methods are pivotal, as they reflect distinct underlying objectives in the modeling process.

The process of fitting data involves not only aligning with known data but also anticipating future, unknown data in forecasts [106, 273]. To do that, two different approaches are possible. Knowledge can be incorporated within a model as physical or semi-physical equations [130] or with the support of historical data collected in the past. Despite recent technological advancements covering both of these cases, academia and industry are predominantly investing time and resources on the second option. This preference stems from the fact that the completely data-driven, and in some extreme cases, black box approach of Artificial Intelligence has already outperformed humans in certain highly specific tasks.

As a consequence, Informed AI [263] describes the injection of prior knowledge into learning systems. Such knowledge often comes from domain-expert's prior knowledge especially in scientific and engineering domains. In robotics, simulation tools like ISAAC GYM [157], which heavily rely on physical equations, are now used to generate synthetic data for AI-oriented training processes when the available datasets are insufficient for Deep Reinforcement Learning applications. In neural networks for climate prediction, physical laws are incorporated using knowledge-based loss functions, illustrating the effectiveness of Informed AI for temperature modeling [66]. In this approach, physical relationships between temperature, density, and depth of water are utilized to design a physics-based loss function. The same technique has been proposed in Autonomous Driving scenarios [274] leveraging additional, already existing sources of knowledge is key to overcome the limitations of purely data-driven approaches.

In a different fashion, Generative AI [80] aims to solve the same lack of information. StyleGAN [139] is a generative adversarial network (GAN) capable of unsupervised separation of high-level attributes (e.g., pose and identity when trained on human faces) and stochastic variation in the generated images (e.g., freckles, hair) allowing the automated generation of high-quality dataset.

While both Informed and Generative AI solves the scarcity of data problem in learning pipelines, they do not minimize the computation required to train the final model. Actually, the overall cost of the process is instead increased since an additional step is required at the beginning of the pipeline. Ensemble is a general way of improving the accuracy and stability of learning models, especially for the generalization ability on small datasets. The main idea of a typical ensemble classification model consists of two steps: (i) generating classification results using multiple weak classifiers, and (ii) integrating multiple results into a consistency function to get the final result with voting schemes. Bagging [39], AdaBoost [111], random forest [40], random subspace [118] and gradient boosting [86] are just some of the widely-used ensemble classification methods.

The authors of [17] approach the issue of assessing the uncertainty forecasts of climate impacts on river streamflow and provide an impressive example of ensemble learning in climate change prediction. In comparison to 1964–1990, they examined ensemble estimates of hydrological changes in the Alpine Rhine (Eastern Switzerland) for the short- and long-term scenario years 2024–2050 and 2073–2099. Dzeroski et al. [77] explored the creation of ensembles composed of heterogeneous classifiers using stacking. They demonstrated that, at best, these ensembles perform comparably to selecting the best classifier from the ensemble through cross-validation. The researchers introduced two novel stacking methods by extending the approach with probability distributions and multiresponse linear regression. Their findings indicated that the latter extension outperforms existing stacking approaches and even surpasses the performance of selecting the best classifier through cross-validation.

In [189] authors suggest the integration of the Ensemble Model Output Statistics (EMOS), a state-of-the-art technique, with an ensemble that undergoes adjustment through an autoregressive process fitted to the corresponding error series. This adjustment is achieved via a spread-adjusted linear pool, specifically designed for temperature forecasts. The introduced ensemble modification technique serves the dual purpose of adjusting the ensemble directly and obtaining a comprehensive predictive distribution for the weather quantity. The experimental results, demonstrated using temperature forecasts from the European Centre for Medium-Range Weather Forecasts ensemble, showcase the effectiveness of the proposed approach in yielding improved results compared to the basic (local) EMOS method. Pado et al. [204] present a novel ensemble methodology for predicting long-term energy demand, integrating various models such as auto-regressive integrated moving average, artificial neural network, and others. Through rigorous comparisons, the proposed approach demonstrates a 22.3% decrease in mean squared error and a 33.1% reduction in mean absolute percentage error compared to leading models.

Employing autoregressive techniques, authors in [22] propose an ensemble of ARIMA models [239] to predict the annual energy consumption in Iran. The results demonstrate that the proposed hybrid patterns enhance the accuracy of single ARIMA model in predicting energy consumption. Shahriari et al. [234] addresses the challenge of traffic volume prediction, comparing non-parametric and parametric methods. While parametric methods, like ARIMA, exhibit low accuracy, non-parametric methods are criticized for lacking theoretical support. Their innovation lies in combining bootstrap with the ARIMA model, resulting in an ensemble of ARIMA models (E-ARIMA) to enhance prediction accuracy while maintaining theoretical adherence.

Specifically for traffic forecasting, the study conducted by Vinay et al. [95] focuses on enhancing public transportation systems through improved road safety and traffic circulation using Vehicular Ad-Hoc Networks and Intelligent Transportation Systems. The paper discusses the application of ARIMA techniques, emphasizing the integration of short-range and long-range dependencies in historical traffic volume data. The analysis considers various types of roads and evaluates the computational complexity of ARIMA. Empirical results reveal that ARIMA-GARCH outperforms ARIMA and SARIMA [49] in road traffic prediction, demonstrating stability in model order across different historical traffic volumes and types of roads. Yao et al. [287] introduce a Spatial-Temporal Dynamic Network (STDN) to model the complex spatial dependencies and temporal dynamics in teh city of New York. The empirical findings from a real-world dataset reveal that the temporal dependency exhibits a daily and weekly pattern; however, it is characterized by dynamic temporal shifting rather than strictly adhering to periodic behavior.

In the following section, we novelly propose a blockchain based framework for improving trust among FL participants. Subsequently, we address the trade-off between model complexity and dataset size showcasing the superior performance of our weak autoregressors ensembling in a real-world time series forecasting use case.

4.2 Blockchain-based Trustworthy Federated Learning

To enhance the overall trustworthiness of the FL process, we introduced TruFlaaS [175], a framework designed to establish trust among third-party contributors to the FL process. TruFlaaS introduces a novel validation strategy for aggregating partial models, leading to an enhanced quality of the global model. Initially, we assign a trust level to each client, updated in each round, to appropriately weigh their contributions.

The blockchain, smart contracts, validation set, and Decentralized Oracle Networks (DON) form the key architectural components that contribute to achieving a trustworthy Federated Learning as a Service (FLaaS) framework. A DON serves as a middleware layer facilitating the secure and reliable delivery of off-chain validation data to the blockchain. The integration of blockchain and smart contracts enhances participant trust in the Federated Learning (FL) process. Specifically, a smart contract validates partial models using a validation set provided by the DON. Subsequently, it aggregates contributions from clients, assigning weights based on their level of trustworthiness. Honest participants are incentivized to engage, while malicious adversaries face discouragement through penalization mechanisms. TruFLaaS exhibits the flexibility required to cater to diverse client demands. To our knowledge, TruFLaaS stands out as the first framework designed and implemented to instill trustworthiness in the FLaaS paradigm, incorporating smart contracts and a DON for partial model validation in Federated Learning. The main components of the system are (see Fig. 4.1 as a reference):

- **Decentralized Oracle Network (DON)**, it is the most innovative component on the Service Provider side. This serves as an intermediary layer between the service provider and the blockchain. This architecture offers several advantages. Without a DON, validation data would need to be directly published on the blockchain. However, this approach has a downside, as all participants could potentially exploit the published validation data to create a partial model, even if malicious, that successfully passes the validation phase. To maintain the integrity and security of the Federated Learning (FL) process, it is crucial that validation data is provided only after the prerequisites for aggregation have been satisfied. For instance, in the context of predictive maintenance, a malicious client might attempt to construct a partial model that performs well on validation data but fails to accurately estimate the remaining useful lifetime (RUL) when it falls below a certain threshold.

- **Blockchain Node**, the service provider must deploy blockchain nodes to guarantee authentication and authorization. Clients have the option to either locally run a blockchain node (as depicted in Fig. 4.1) or connect to one deployed by the service provider. The choice between running a blockchain node locally and connecting to a proxy node hosted by the service provider involves trade-offs. While running a local blockchain node provides clients with direct visibility into FL processes, it may consume a considerable amount of client resources. This could pose a challenge for clients with limited computing power or storage capacity. Thus, the decision to run a blockchain node or connect to a proxy node needs to carefully consider these trade-offs, allowing for a flexible configuration that balances ease of use and resource consumption in FL.

- **Validation set**, the validation and aggregation of the global model are orchestrated through a smart contract. The smart contract initiates by verifying the authorization status of the client to participate in the current FL process. Subsequently, before validating and aggregating partial models, the smart contract patiently waits until the specified training requirements are met. For instance, if the aggregation strategy mandates that all participants must contribute their updates, the smart contract remains in a waiting state until all clients have submitted their partial models. It then triggers a request for the validation set for the given round. As mentioned earlier, the validation set is supplied by the service provider through a DON. Each partial model undergoes validation against the acquired validation set, and those meeting the predetermined performance threshold are incorporated into the aggregation phase.

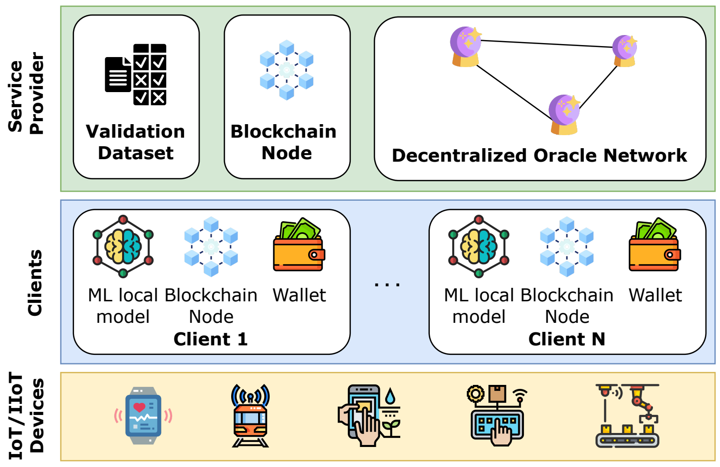


Figure 4.1: Trustworthy Federated Learning as a Service architecture.

Determining the Trust Level of Participating Nodes:

Each client, denoted as $c_i \in \mathscr{C}$, is attributed a trust level $t_{i,l} \in [0,1]$. Trust models in Peer-to-Peer (P2P) networks often differentiate trust towards a peer into direct and indirect categories. Direct trust is grounded in prior interactions with that peer, while indirect trust is based on the peer's global reputation. The *Transaction Acceptance Rate* ($TAR_{i,l}$) is introduced as a measure, defined as:

$$TAR_{i,l} = \frac{TA_{i,l}}{T_{i,l}}$$

where $TA_{i,l}$ is the number of accepted transactions, and $T_{i,l}$ is the total number of transactions related to the f_l process. The *Global Trust Value* (GT_i) is computed as:

$$GT_i = \frac{\sum_{j=1}^{N} t_{i,j}}{N}$$

Here, trust levels are derived from clients' participation in past or ongoing Federated Learning (FL) processes. By leveraging both direct and indirect trust components, the trust level is calculated using the formula:

$$t_{i,l} = \frac{GT_i + TAR_{i,l}}{2}$$

These values undergo updates at each round through the dedicated smart contract (Fig. 4.2, step 11). At this juncture, a decision is made on whether to revoke the $vc_{i,l}$ from client c_i if its $TAR_{i,l}$ fails to meet minimum requirements (i.e., falls below a predefined threshold). In each round, the average TAR (μ_{TAR}) across all participants and the corresponding standard deviation

(σ_{TAR}) are computed. It is then assessed whether, considering the number of remaining rounds, a client can surpass the threshold set at $\mu_{TAR} - \sigma_{TAR}$. If it fails to do so, the corresponding $vc_{i,l}$ is revoked, and the client is excluded from f_l.

The reward $r_{i,l}$ assigned to each c_i is then calculated as:

$$r_{i,l} = b_l \frac{TAR_{i,l}}{\sum_{j=1}^{N} TAR_{j,l}} \tag{4.1}$$

Such an incentive scheme fairly distributes b_l according to the contributions of all the $c_{i,l} \in \mathscr{C}_l$. For each $c_{i,l}$, the contribution corresponds to its $TAR_{i,l}$. It is clear that, given $c_{i,l}, c_{j,l} \in \mathscr{C}_l$, and $TAR_{i,l} > TAR_{j,l}$, it follows that $r_{i,l} > r_{j,l}$.

Step-by-step Validation and Aggregation Process:

After initiating a new f_l or joining an existing one, a client is furnished with the requisite $vc_{i,l}$ to contribute to that training. The validation and aggregation workflow, depicted in Fig. 4.2, involves the following steps, iterated for each round k:

1. Each $c_{i,l} \in \mathscr{C}_l$ collects local data from deployed IoT/IIoT devices.
2. The data are employed to locally train a partial model $mp_{i,l}^k$.
3. Each $c_{i,l}$ provides $mp_{i,l}^k$ and $vp_{i,l}$, obtained by signing $vc_{i,l}$ through its Decentralized Identifier (DID), to sc_l, which validates and aggregates all partial models MP_l^k.
4. sc_l forwards $vp_{i,l}$ to a smart contract responsible for authorizing participants.
5. This smart contract grants or denies access to f_l. It jointly verifies the validity of $vp_{i,l}$ and ensures that the embedded $vc_{i,l}$ has not been revoked.
6. Before aggregating all partial models MP_l^k, sc_l awaits until the aggregation requirements are met and validates MP_l^k against a validation set $V_l^k \subset \mathscr{V}_l$ provided by a Decentralized Oracle Network (DON) d.
7. d requests V_l^k from s.
8. s provides V_l^k to d. Ensuring that V_l^z is distinct from V_l^j for $j < z$ prevents $c_{i,l}$ from crafting mp_i^z to achieve satisfactory performance on a known V_l [].
9. d returns V_l^k to sc_l.

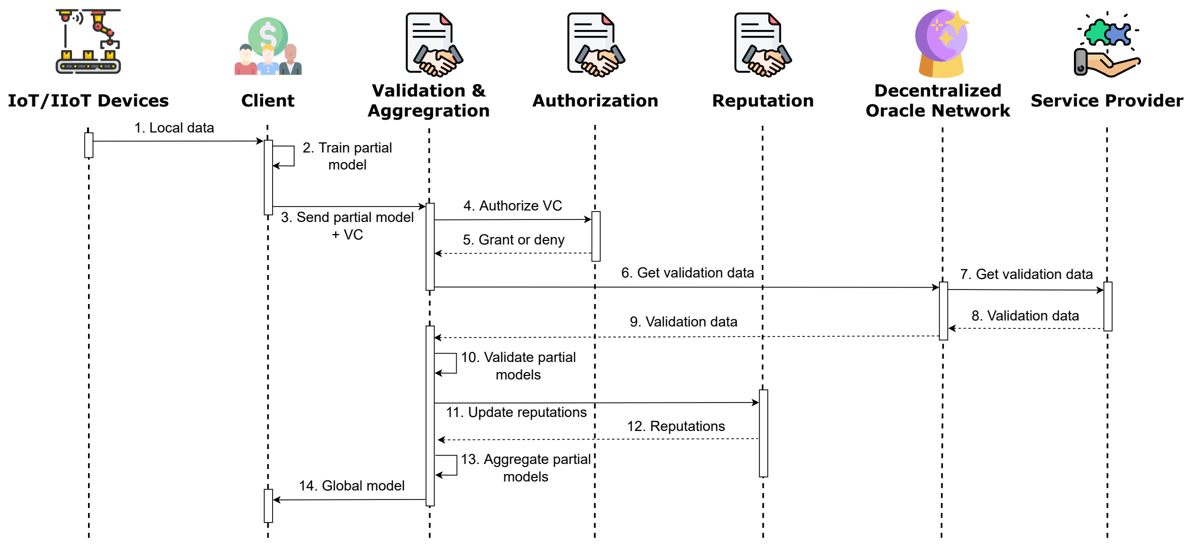


Figure 4.2: Validation and Aggregation process inside True-FlaaS

10. sc_l validates MP_l^k against V_l^k. To be accepted, a partial model must achieve performance equal to or surpass a specific threshold. The *interquartile range* (IQR) method is employed for outlier detection, offering robustness to extreme values.

11. Based on collected metrics, sc_l sends updated reputations to a smart contract used to trace t_i for each c_i.

12. The smart contracts calculate all trust levels T^k for each $c_{i,l}$ and return them to sc_l.

13. sc_l aggregates all validated $mp_i^k \in MP_l^k$, weighting them according to the corresponding $t_i^k \in T^k$.

14. The global model mg_l^k is provided to all $c_{i,l}$.

Experiments and Corresponding Results:

To prove the effectiveness of True-FlaaS in real-world scenarios we tested our middleware in two different industrial context: (i) the predictive maintenance use case, employing the NASA Turbofan Jet Engine dataset [229] for engine degradation modeling and; (ii) the botnet attack detection use case where we used the N-BaIoT dataset [178] containing real traffic data gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE. We conducted multiple experiments, considering scenarios involving both honest clients with limited datasets and malicious nodes. The malicious nodes had intentions to disrupt the training process or introduce backdoors within the global model. Our evaluation included a comparison of TruFLaaS against two counterparts: the conventional baseline (without validation mechanisms) and TrustFed [213].

For each of the two datasets, we conducted three different sets of experiments. Initially, we considered a scenario where the data distribution among honest clients was heterogeneous, meaning that some nodes possessed more or fewer data samples than others. These experiments were performed by varying the number of participants and incorporating different percentages of nodes with varying data amounts.

Secondly, we delved into a specific case within the aforementioned scenario, where some nodes lacked data for specific classes of events, defined as "rare cases." For instance, certain smart manufacturing enterprises may not have experienced the breakdown of specific machinery or have never been affected by particular types of attacks. In this experiment, focusing on the predictive maintenance use case, we identified rare records based on the Remaining Useful Life (RUL) values. We discriminated data records by tagging them as rare if they fell within a low percentile of a pseudo-normal distribution, separating low RUL values from the others. To accomplish this, we conducted statistical analysis and calculated the 10th percentile values for both the training and validation sets. Given that NASA data do not follow a normal distribution, we standardized the data to use the z-score table for calculating exact areas for any given normally distributed populations.

Lastly, to evaluate the resilience of TruFLaaS against backdoor attacks, we performed several experiments with varying percentages of malicious nodes. Similar to TrustFed, we simulated the behavior of malicious nodes in these experiments by performing training on data containing random noise.

For the heterogeneous data distribution experiments, TrustFed, selectively aggregates partial models whose accuracy falls within a predefined interval determined by the neighborhood of the mean and standard deviation. However, this approach may overlook clients with heterogeneous data distributions that produce high-quality partial models, surpassing the bounds of the specified interval. In contrast, TruFLaaS discards partial models below its accuracy threshold during validation but involves all partial models with accuracy exceeding the lower bound of the mean and standard deviation during the aggregation phase. Figures 4.5 and 4.6 illustrate how TruFLaaS surpasses TrustFed by effectively identifying client contributions with significantly larger datasets. TruFLaaS not only achieves the target accuracy faster than other approaches but also exhibits a more pronounced advantage with an increasing number of nodes possessing augmented data. In both cases experiments encompass accuracy comparison of different node selection strategies with 0 nodes having augmented data (a), 10 nodes having augmented data (b), and 25 nodes having augmented data (c).

In the Rare Cases scenario, for the sake of fairness, we did not compare TruFLaaS with TrustFed, as the latter does not differentiate handling rare cases, and our approach is expected to outperform it by far. Figures 4.5 and 4.6 present the experimental results, considering variations in the number of nodes without rare data and the strategy employed to exclude nodes with subpar performance. In the predictive maintenance scenario, the first two strategies, i.e., excluding nodes with poor performance on the validation set comprising only rare cases and excluding nodes with poor performance on the validation set mirroring the distribution of the test set, exhibit superior performance compared to other aggregation strategies. For the botnet attack scenario, our solution demonstrates resilience to an increasing number of nodes lacking rare data. Notably, the accuracy remains unaffected even with a higher proportion of nodes with heterogeneous datasets. All experiments conver accuracy comparison of different node selection strategies with 0 nodes without rare data (a), 10 nodes without rare data (b), and 25 nodes without rare data (c).

For the Model Forging Attack set of experiments, Figures 4.7 and 4.8 vividly illustrate how TruFLaaS exhibits greater robustness than TrustFed against model forging attacks while varying the number of malicious nodes. This robustness stems from the fact that TrustFed, relying on mean and standard deviation for outlier detection, becomes less accurate in the presence of excessively large or small values. For instance, an outlier with notably poor performance might significantly skew the total mean, leading TrustFed to accept outliers whose performance should not be accepted under normal conditions. Furthermore, TrustFed's approach, which does not weigh partial models, results in the complete disruption of training when aggregating an outlier. Notably, in the predictive maintenance experiment, TrustFed performs worse than the baseline with 0 malicious nodes, possibly due to the removal of nodes achieving performances much greater than the average. Conversely, TruFLaaS utilizes a more robust outlier detection algorithm, consistently achieving similar performance across all scenarios. Additionally, by employing weights based on the number of accepted transactions (i.e., level of trust), sporadic errors during the validation process have minimal influence on the global model, preventing disruption of the entire training process. These results provide valuable insights into the robustness of our proposal, demonstrating its effectiveness in protecting against model forging attacks.

Heterogeneous Data Distribution results:

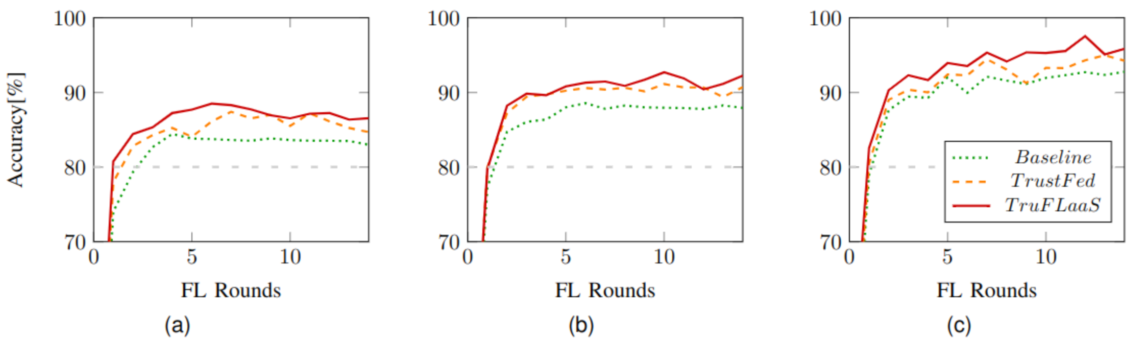


Figure 4.3: Heterogeneous Data Distribution on NASA engine degradation dataset.

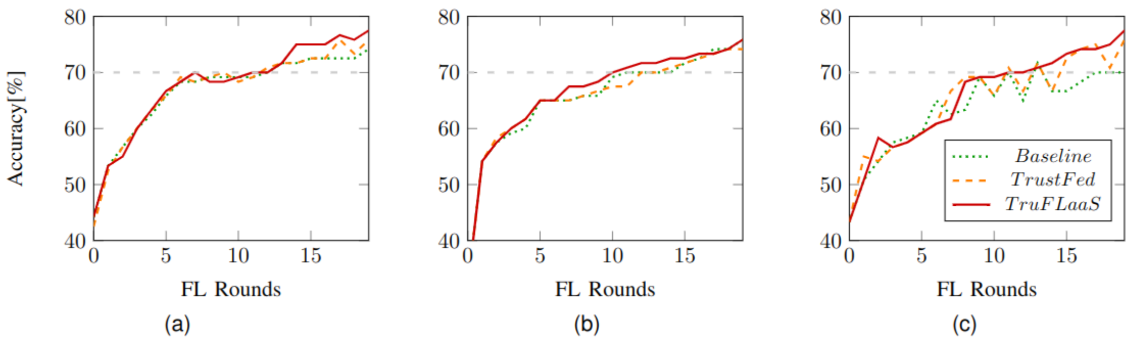


Figure 4.4: Heterogeneous Data Distribution on N-BaIoT botnet attack dataset.

Heterogeneous Data Distribution on Rare Cases results:

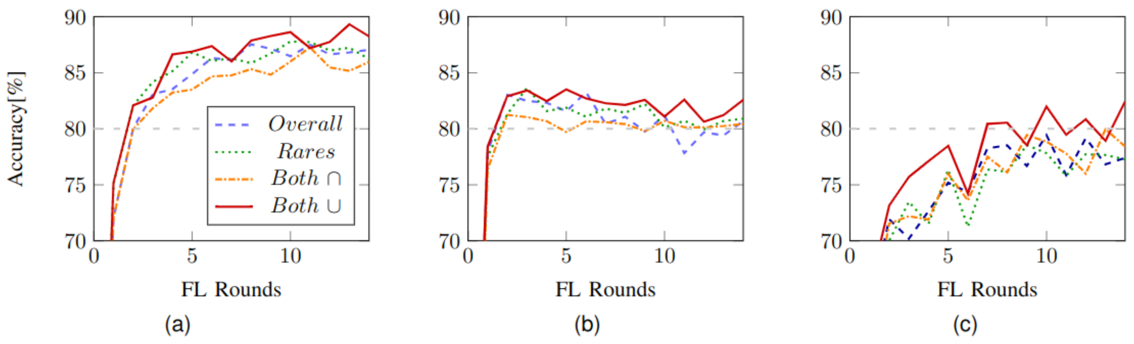


Figure 4.5: Heterogeneous Data Distribution on Rare Cases over NASA engine degradation dataset.

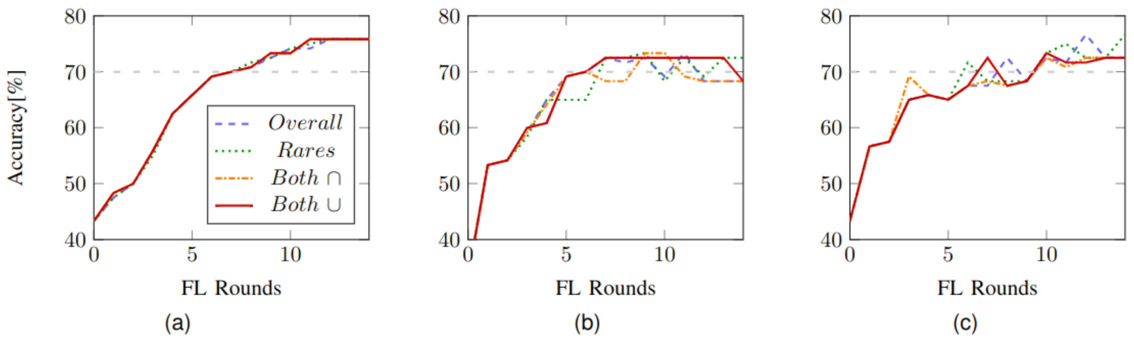


Figure 4.6: Heterogeneous Data Distribution on Rare Cases over N-BaIoT botnet attack dataset.

Model Forging Attack results:

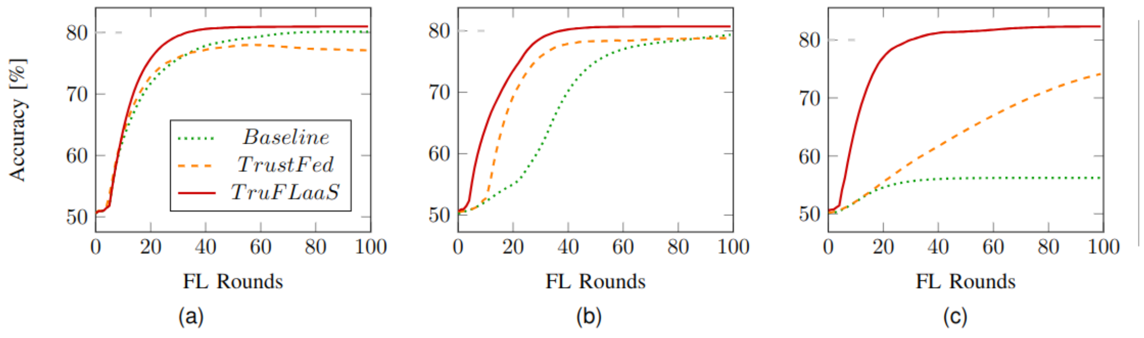


Figure 4.7: Model Forging Attack on NASA engine degradation dataset.

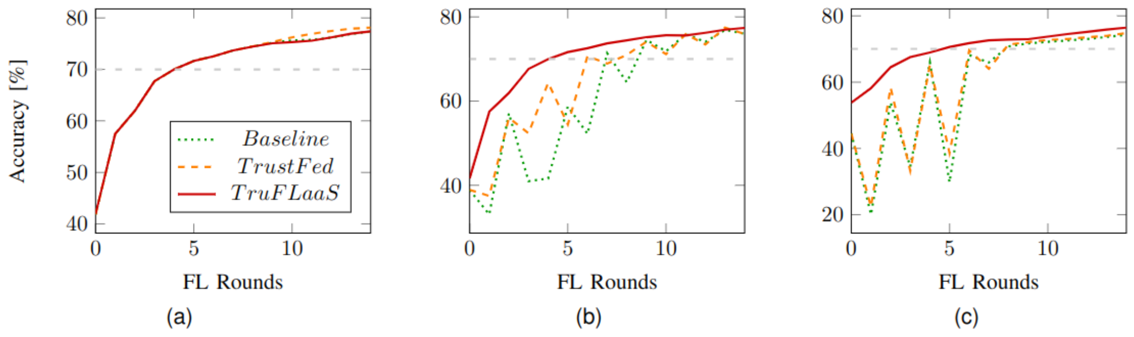


Figure 4.8: Model Forging Attack on N-BaIoT botnet attack dataset.

4.3 Modeling Vehicular Traffic in UFM Scenarios Through Frugal Autoregression

The rapid deployment of sensing and communication technologies is facilitating the implementation of smart city applications, ushering in an era of data-driven modeling for various urban processes. Among these, Urban Facility Management (UFM) has gained prominence for its direct influence on the sustainability and development of cities.

The study we conducted in [181] deals with the predictive subsystem in charge of traffic volume forecasting. This is responsible for generating near-to-mid term forecasts of the activity index, providing UFM operators with a versatile decision-support system. In this context, we present an analysis of the vehicular traffic component, a key element of the activity index, evaluating the accuracy of various predictive models and discussing their operational implications. In this context, we present different prediction approaches, aimed at inferring autocorrelations in the vehicular data by applying and comparing canonical statistic approaches, e.g., Weighted Moving Average (WMA), autoregression techniques, e.g., AutoRegressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA), and deep learning enhanced regression models like Long Short Term Memory (LSTM) [119]. We assess the accuracy of the schemes using realistic data, discussing the tradeoffs and operational considerations that emerge. In particular, we show how the combination of a data segmentation approach and ARIMA outperforms both WMA and LSTM deep learning regression by about 10% of accuracy score while achieving comparable computation time.

A Real World Use Case:

The historical dataset covers a period from April 2019 to June 2020 in the city of Bologna (IT), including the first lockdown phase announced in Italy (March 2020). The data are sourced by an onboard blackbox, a multi-purpose and autonomous device, when pre-determined events occur, e.g., vehicle engine turned on/off. The historical data comprises daily trips, delimited by a start and stop latitude/longitude coordinate, along with some additional attributes used to qualify a trip as follows:

- **Trip and device identifier:** numerical identifier of the trip and of the on-board device, respectively.

- **Start/End date/time:** time/date when this trip was initiated i.e., corresponding to the time/date the vehicle engine is turned on/off.

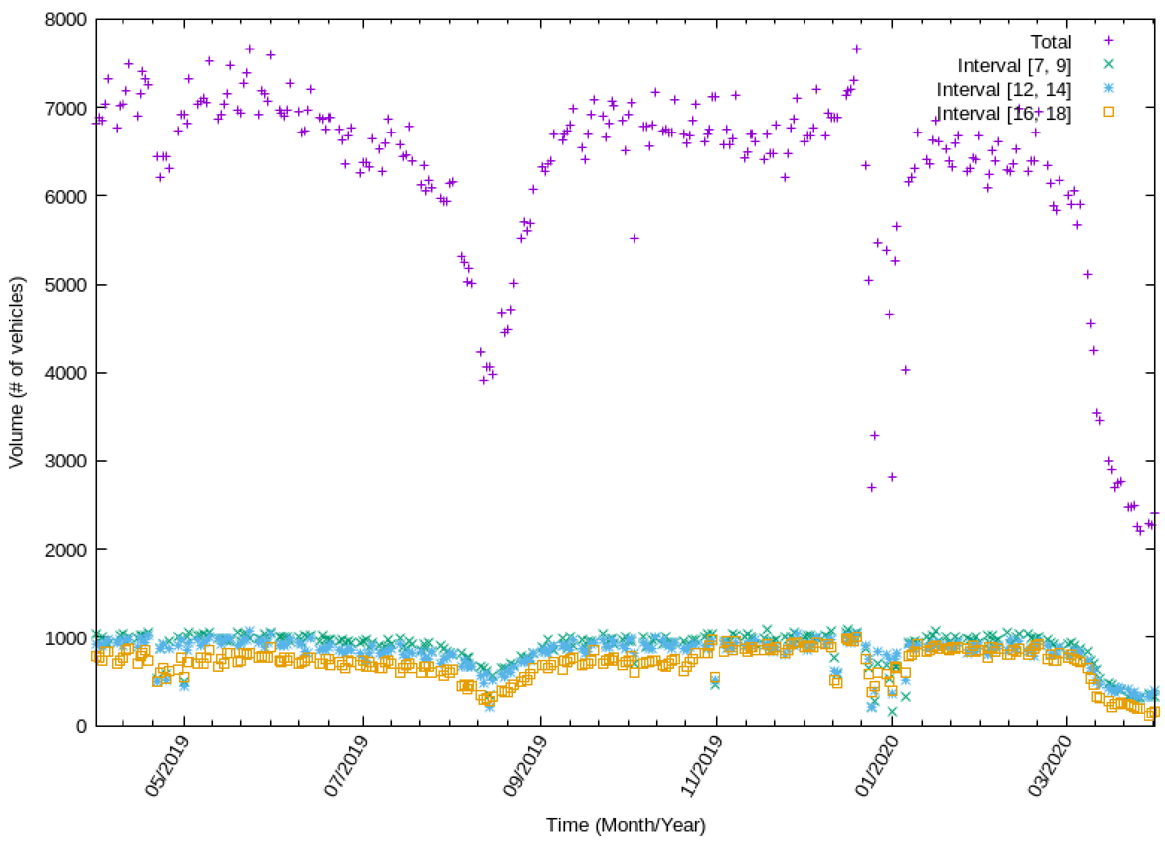


Figure 4.9: Daily number of trips evolution in time.

- **Start/End latitude/longitude:** GPS coordinates denoting the place when this trip initiated/terminated.

- **Average velocity:** average speed in *km*/*h* of the trip.

In addition to the trip header data described above, in a separate file, the system receives and digests intermediary, latitude/longitude, data points denoting intermediary trip events sensed by the on-board device. In general, an intermediate data point is generated whenever a vehicle has traversed $1km$ or $60seconds$ have passed since the last data generation.

The graph presented in Figure 4.9 depicts the daily trip volume derived from the dataset, showcasing its temporal evolution. Utilizing the trip identifier within the datasets facilitates the extraction of this information. The graph provides insights into the overall volume of trips, further broken down into three distinct time intervals: 7-9 am, 12-2 pm, and 4-6 pm. Notably, the sudden decline in traffic volume, commencing in March coinciding with the initiation of lockdown measures following the first decree, is evident from the visualization.

In our pursuit of an effective predictive model with scalability and operational value for Urban Facility Management (UFM) operators, we have developed distinct algorithms to cater to various operational needs:

- **MTPM (Midterm Predictive Model):** This model allows operators to schedule and monitor future maintenance interventions in the midterm;
- **STPM (Short-Term Predictive Model):** Designed for short-term predictions, this model enables operators to (re)schedule planned interventions promptly in response to unexpected interferences in activity.

These approaches aim to deliver reasonably accurate predictions at different geographical granularities, covering aspects such as the entire city road network or the traffic volumes on a specific street. The system tracks data at a road-arch granularity allowing fine grained analysis and forecasting on high and low traffic volumes areas of interest. Operationally, MTPM utilizes 11 historical months to predict the next one, while STPM is trained over the last week to predict the trend for the next day. The algorithms have been tested both with and without weekend days, yielding similar results in both scenarios.

Complexity Analysis Between Autoregression and DL-based Forecasting Techniques:

Regarding the choice of the algorithm to use, we considered ARIMA (Autoregressive Moving Average), SARIMA (Seasonal Autoregressive Moving Average), and LSTM (Long Short-Term Memory).

In terms of computational complexity, ARIMA is the less complex of these approaches [95]. Formally, its computation can be expressed as:

$$O(ARIMA) = \mathscr{O}((N-p)\times p^2 + (N-q)\times q^2) \tag{4.2}$$

Where:

- N is proportional to the dataset size;
- p and q are the two parameters of Moving Average and Autoregression, respectively.

In SARIMA, this is further added to by the periodicity component of the model, bringing the overall complexity to:

$$O(SARIMA) = \mathscr{O}(\sum_{i=1}^{L_s}[(\frac{N}{S_i} - p_i)\times p_i^2 + (\frac{N}{S_i} - p_i)\times q_i^2] + (N-p)\times p^2 + (N-q)\times q^2) \tag{4.3}$$

Where S_i denotes the seasonality component.

Finally, regarding LSTM, being a deep neural network, its complexity is not directly linked to the data distribution but rather to the depth of the network's architecture and the number of parameters to consider in regression. Mathematically, the complexity of a single sample can be described as:

$$O(LSTM) = \mathcal{O}(\sum_{i=1}^{I}(n_{c,i-1} \times s_{c,i}^2 \times m_{c,i}^2)) \tag{4.4}$$

- I: Number of convolutional layers;
- nc: Number of convolutional kernels;
- $nc, i-1$: Proportional to the set of input variables;
- sc, i: Size of the convolutional kernel;
- mc, i: Proportional to the set of output variables.

Specifically in terms of complexity on the vehicular traffic dataset in the city area of Bologna, it has been observed that

$$10 \times O(ARIMA) \approx 4O(SARIMA) \times \approx O(LSTM) \tag{4.5}$$

Autocorrelation Analysis and Experimental Results:

Autocorrelation is commonly used in discrete time series to compare a signal with a delayed copy of itself. This kind of analysis is the mathematical tool for finding repeating patterns which are often obscured by noise or by outliers. Both MTPM and STPM employ an autocorrelation strategy to predict over the aggregated and discrete data, respectively. We first checked the the stationarity of dataset. In fact, ARIMA and SARIMA imply the stationarity hypothesis to converge on time series values. The autocorrelation function (ACF) assesses the correlation between observations in a time series for a set of lags and helps to determine if a function is stationary or not. A lag is the window that has to be considered when computing the autocorrelation value between two different samples. If a significant lag exists, then the function presents a relevant pattern between the records at the same index of the lag value. On the contrary, if no relevant ACF value exists, the function is not stationary and no autogressor can be applied.

As shown in Fig. 4.10, ACF identifies 7 and its multiples as significant and positive lags outside the confidence interval, hence the function is not stationary and neither autoregressive nor moving average method can be applied. In this case, a common approach is to differentiate

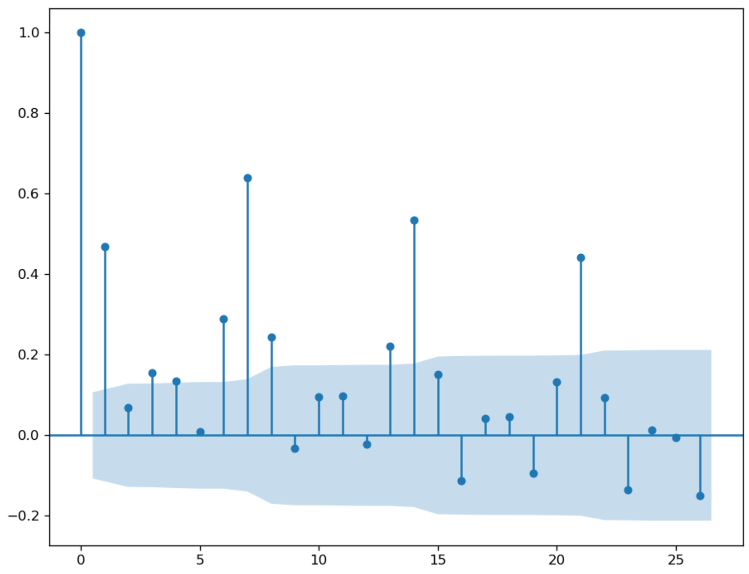


Figure 4.10: Graphical evaluation of ACF. Consistent recurrence of 7 and its multiples as lag values in the autocorrelation function.

the function multiple times to change its time dependency, deleting the trend. The novelty of the work done is that instead of differentiating the function, the data are split according to the dependency between the same day of the week during the whole time series. In other words, weekdays were grouped to have all the Mondays, Tuesdays, Wednesdays and so on, together. Both the aggregated and the fine-grained (road) data were split according to the this data segmentation strategy.

Then we evaluated model performance on aggregated data across ARIMA, SARIMA and LSTM. The accuracy of the devised algorithms is measured by calculating the *Mean Absolute Percentage Error* (MAPE) to take into account the magnitude of data in the residuals' computation. Table 4.1 reports a day by day comparison between considered approached highlighting ARIMA superior performance after segmenting the dataset into da of the week subsets.

We then extended our evaluation to fine-grained data. In this case we selected two stress of the city in order to compare the performance of the best mode, ARIMA, in different scenarios. In the first one we selected a street with a corresponding high data volume - 2000 vehicles a day - and a secondary road with about 15 vehicles a day. As reported in the aggregated data scenario, ARIMA outperforms SARIMA in all weekdays, with an overall percentage accuracy of 83.39% for ARIMA and 48.32% for SARIMA. Another relevant consideration regards the performance of MTPM over high and low volume traffic roads. Table 4.2 shows that ARIMAs' percentage accuracy on high traffic streets is on average a third higher when compared to the low traffic streets. This is reasonable, as the higher the data volume, the smaller the impact of noise and

Day	ARIMA	SARIMA	LSTM
Mondays	94.78%	92.70%	91.63%
Tuesdays	97.94%	81.01%	73.56%
Wednesdays	96.45%	33.57%	78.21%
Thursdays	94.63%	75.99%	54.87%
Fridays	94.64%	75.97%	64.14%
Saturdays	94.24%	92.70%	83.47%
Sundays	91.51%	90.93%	87.38%

Table 4.1: ARIMA/SARIMA/LSTM accuracy: day of week comparison

outliers on the prediction when calculating the MAPE metric. In fact, the standard deviation for high traffic street is much higher than the one for low traffic streets. More specifically, they are 105.98 and 5.04, for high and low traffic volume roads, respectively.

ARIMA		
Day	High traffic	Low traffic
Mondays	92.01%	54.25%
Tuesdays	91.96%	65.56%
Wednesdays	96.99%	72.36%
Thursdays	97.05%	78.31%
Fridays	98.64%	75.31%
Saturdays	95.66%	74.35%
Sundays	95.09%	78.47%

Table 4.2: ARIMA/SARIMA/LSTM accuracy: ARIMA percentage accuracy comparison between high and low traffic volume roads.

Figure 4.2 illustrates the performance of ARIMA over SARIMA in a high(a) and low(b) traffic scenario. As reported, ARIMA follows the trend with a reasonable accuracy, well placed in the middle of the confidence interval. On the contrary, SARIMA predicts an increasing trend which is not observed in the raw data.

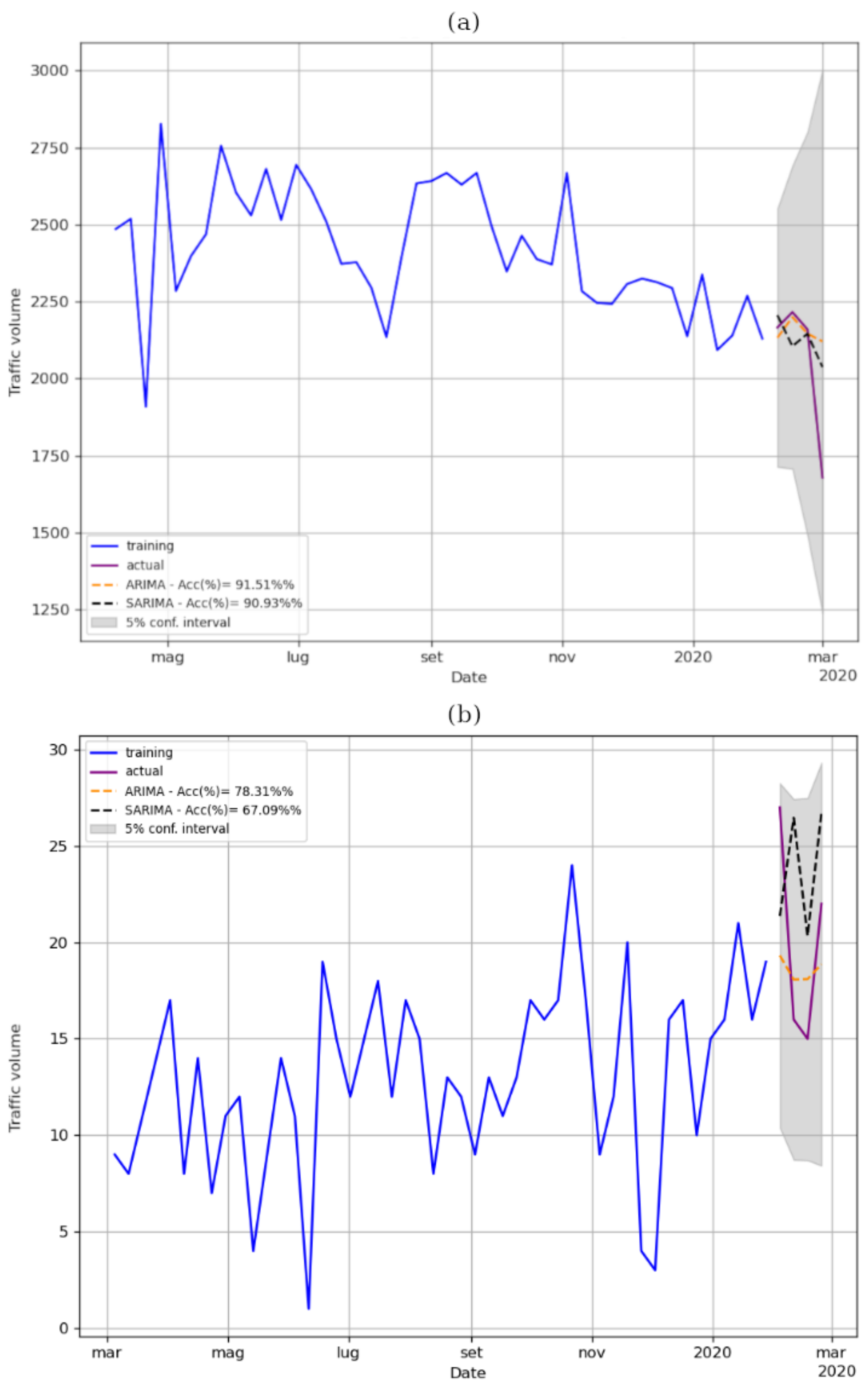


Figure 4.11: Models' prediction comparison on (a) high and (b) low traffic volume streets.

Chapter 5

Communication Efficient Frugality for Bandwidth-Constrained Environments

In the realm of Mobile Computing, the growing demand for mobile devices to run high performance applications, coupled with the need to extend the battery life of handheld mobile devices [147], highlights the importance of alternative computing strategies. Offloading emerges as a powerful approach to prolong the operational duration of handheld mobile devices by delegating certain application components to remote execution locations, such as servers in data centers or cloud environments. The main concern in this case focuses on issues of relatively high latency for applications sensitive to delays. Computation offloading encounters challenges, particularly in the context of data migration among components. These challenges are often exacerbated by wireless communication, which emerges as a dominant factor influencing the overall execution time of applications. The time required to transfer data between a mobile device and the server via a wireless link can significantly impact the overall execution time, potentially causing deviations from time constraints. Therefore, the data transmission rate between the mobile device and the server emerges as a critical factor influencing offloading decisions in such scenarios.

Additionally, unreliable networking conditions are very likely to take place once node mobility comes into play, introducing further complexities in maintaining stable and efficient communication between nodes in a mobile environment. This limits the applicability of offloading to those environments where networking conditions are stable and predictable. As a consequence it is often preferable to move the entire computational workload on mobile devices leading to additional power constraints, as we already mentioned in Section 3.1.

In this section, we thoroughly delve into the various aspects of the offloading paradigm, aiming to attain communication frugality in the face of challenging bandwidth requirements.

Specifically, in Section 5.1, we scrutinize existing solutions utilized to surmount networking constraints. In Section 5.2, we present Furcifer, a container-based framework designed for semantic compression in Object Detection tasks.

5.1 Leveraging Networking Conditions As Opportunistic Frugal Resources

Here we consider bandwidth as another limited resource available to mobile devices, alongside energy and data constrains as done in Chapter 3 and Chapter 4, respectively. Local Computing (*LC*) and Edge Computing (*EC*) stand as the primary pragmatic strategies for tackling the wide array of heterogeneous tasks in the broad range of real-world scenarios centered on the execution of complex data analysis and decision making algorithms. On the one hand, *LC*, that is, the execution of the ML algorithm onboard a mobile device, investigates the optimal symbiotic interplay between software applications and hardware components, aiming to harness the available onboard resources in the utmost efficient manner possible. On the other hand, *EC*, where the ML tasks are offloaded to a compute-capable device positioned at the network's edge, leverages high-performance communication and computing technologies to effectively support real-time applications. While *EC* promises higher computational capabilities and lower energy consumption to mobile systems, it requires a high capacity wireless link that may be not consistently available due to the volatile nature of wireless channels, resulting in unreliable computing services. Conversely, *LC* is a more reliable computing modality with near-deterministic performance. However, the deployment of ML models on mobile devices comes at the price of reduced lifetime due to high energy consumption and diminished performance due to limited onboard resources.

Optimizing Full Edge Offloading: SOTA Approaches and Applications

Authors in [124] introduce an energy-efficient approach for meeting application execution time requirements by employing a dynamic offloading algorithm based on Lyapunov optimization [69]. The algorithm demonstrates low complexity in solving the offloading problem, determining the software components to execute remotely based on available wireless network connectivity. Performance evaluations indicate that the proposed algorithm outperforms existing methods, achieving energy savings while meeting application execution time constraints. Additionally, to address diverse wireless conditions and ensure low-latency user interaction, we propose an adaptive offloading algorithm. This dynamic algorithm leverages Lyapunov optimization to offload specific computation tasks to a dedicated server based on real-time changes in the wireless environment. Comparative assessments show that our algorithm surpasses existing approaches, notably saving more energy, all while adhering to the execution time constraints required by the mobile application.

Chun et al. introduce the CloneCloud framework in [59], designed to enhance both performance and battery life of mobile devices by offloading resource-intensive components to cloud servers. In the framework's partitioning phase, static program analysis and program profiling are combined to identify offloadable components while adhering to specific constraints, such as executing methods that involve mobile sensors locally. The partitioning process operates at the thread level granularity, leveraging static analysis to identify migration constraints and profiling to establish a cost model for offloading and execution. Application partitioning and integration occur at the application level. Upon the user's attempt to launch a partitioned application, the framework consults a database of pre-computed partitions, considering current execution conditions like available network bandwidth and cloud resources. The outcome of this verification is a partition configuration file, loaded by the application binary. This file instruments the selected methods with migration capabilities.

MAUI [61] operates as an offloading framework with a primary objective of optimizing communication between smartphones and cloud services. It stands out as a highly dynamic system due to its continuous profiling process. The framework adeptly conceals the intricacies of remote execution from the mobile user, providing an illusion that the entire application is running on the mobile device. In the initial phase, the MAUI profiler gauges the device characteristics, and it persistently monitors both program and network characteristics throughout the execution. This continuous monitoring is crucial since these characteristics may undergo changes, and outdated or inaccurate measurements could lead to sub-optimal decisions by MAUI. The offloading decisions in MAUI are made dynamically at runtime. The framework determines which components should be executed remotely based on the real-time input from the MAUI profiler. The decision-making process is facilitated by the MAUI solver, ensuring that offloading choices align with the current state and requirements of the device and network.

Zhao et al. introduce the mirror server framework in [294], leveraging Telecommunication Service Provider (TSP) based remote services. TSPs, as communication service providers, primarily offer voice communication services, including landline telephone services. The mirror server framework enhances smartphone capabilities by offering three distinct services: computation offloading, security, and storage. The mirror server itself acts as a robust server equipped with VM templates tailored for various mobile device platforms. A notable characteristic of this framework in contrast with aforementioned contributions is that it does not necessitate application partitioning, opting for the offloading of the entire application. In the preparation phase, a new VM instance, referred to as the mobile mirror, is instantiated. The mirror server takes charge of the management and deployment of these mobile mirrors on a computing infrastructure within the telecom network. Applications run within these mirror VM instances, and the results are subsequently sent back to the Smartphone Mobile Device (SMD). The framework employs an optimized offloading mechanism. However, it is important to note that mirror servers are not specifically designed for extensive data processing. Consequently, they can only provide limited services, such as file caching and file scanning, in contrast to the broader range of services available in cloud data centers.

Xia et al. introduce the Phone2Cloud computation offloading framework in [275], aiming to enhance the energy efficiency of smartphones and improve overall application performance. Distinguishing itself from prior frameworks, this work focuses on a comprehensive quantitative analysis of energy savings through both application and scenario experiments. Phone2Cloud operates as a semi-automatic offloading framework, requiring manual modifications to applications during the preparation step to enable their execution on cloud servers. The offloading decision-making process relies on static analysis, taking into account the user's threshold for delay tolerance. For applications tolerant to delays, the framework employs a straightforward model that anticipates WiFi connectivity. The threshold is determined based on predictions for delaying transfers, allowing more data to be offloaded over WiFi while adhering to the application's tolerance threshold [20]. Notably, the framework opts to wait for WiFi availability (only if 4G savings are anticipated within the application's delay tolerance) rather than immediately initiating data transmission.

In [142], Kemp et al. introduce the Cuckoo framework designed for offloading smartphone applications to a cloud server using a Java stub model. The primary goals of Cuckoo are to enhance mobile performance and dynamic application partitioning. The framework seamlessly integrates the Eclipse development tool with the open-source Android framework. During the partitioning step, Cuckoo leverages the existing activity model in Android to distinguish between intensive and non-intensive components of the application. This activity, responsible for presenting a graphical user interface and binding to services, facilitates the offloading of intensive components to any resource equipped with a Java Virtual Machine (JVM). In the preparation phase, developers are required to write offloadable methods twice—once for local computations and once for remote computations. To aid developers, a programming model is provided, supporting dropped connections, local and remote execution, and packaging all codes into a single compatible remote implementation. Cuckoo operates as a dynamic offloading framework, making decisions at runtime and offloading well-defined components of the application. If remote resources are unreachable (e.g., due to a lack of network connection), the application can seamlessly execute on local resources (the mobile device).

Context Aware Adaptation for Edge Intelligence: SOTA Approaches and Applications

Recent studies have investigated the benefits of context aware adaptation for specific tasks such as 4k mobile Augmented Reality (AR) [214] and mobile video streaming [209] providing an in depth overview of the set of optimization operations required to effectively deploy a self-adaptive policy manager in real-world field experiments, where the level of added complexity increases significantly if compared with theory-oriented setups.

Progressive adaptation to different Edge Intelligence contexts by using synthetic data [120] or uncertainty-aware domain adaptation networks [105] proved to be a promising direction when deploying an ML-based solution trained on state-of-the-art datasets in a real-world scenario. Furthermore, employing strategies such as random exploration of optimal scaling factors [283] can help alleviate the negative effects of source domain bias. An edge-assisted Deep Learning based framework is proposed in [67], emphasizing the collaborative utilization of mmWave radar and cameras to enhance system robustness, particularly in drone applications.

Offloading intelligent tasks to edge servers necessitates the low-latency transfer of information-rich signals over wireless channels, and thus require high data rate wireless transceivers. In[172], Matsubara Y. et. al. perform an analysis of the components of the total inference time in the context of edge-assisted systems, which empirically demonstrates that the wireless channel plays a major role in the end-to-end task latency. 5G communications are classified by the ITU [116] as Reliable Ultra-Low Latency Communications (URLLC), which implies high data rates compared to traditional Wi-Fi (2.4 GHz). Specifically, the 802.11ac standard [240] uses new technologies such as a new operational bandwidth for base stations from 40MHz to 80MHz, Multi User MIMO for simultaneous transmission and reception of packets, and 256-QAM modulation that improves the data rate when applied to *EC* systems. The use of 5G communication technologies has been studied in the context of a wide range of applications.

For instance, Ren et.al [216] and Singh, Kiran et. al [242] demonstrate that collaboration between the mobile, edge and cloud tiers supporting Web AR applications and QoS awareness in cyber-physical systems leads to higher benefits when using 5G communication due to the smaller time to transfer data between tiers. Yuan Z et.al. [290] and Dai C. et. al. [63] propose systems and algorithms to improve *EC* communication for IoV and IoT applications. Although the evaluation of the communication channel is particular to each application, both show that 5G communications enhance the ability of the system to meet the requirements of the ML task. Particularly, Dai C. et. al. [63] compares the IEEE 802.11n and 802.11ac standards and show a reduction in latency by nearly half the time as the number of *EC* requests increases when using the latter.

To reduce the amount of data transferred over the network, such as images, data compression [162] emerges as primary solution. Even constrained devices with very limited computing capabilities are capable of running lossy image compression (e.g. JPEG ISO 15444 standard [208]) methods in real-time, but fail to keep the same response time when applying close-to-lossless [284] techniques. Many neural models for AI applications are commonly trained and evaluated using state-of-the-art datasets. In Computer Vision, for example, COCO2017 [159] and Pascal VOC [82] are some of the most famous ones. However, these evaluations often overlook the significant performance degradation caused by image compression [34, 57, 90, 203], which is inevitable in practical real time *EC* systems.

5.1.1 Semantic Compression for Informed Communication

Rooted in Shannon Information Theory [235], Semantic Compression aims to find the minimal representation of a generic information at LEVEL-B, where similar semantic meaning could be presented with various expressions (some more optimized that others) compared to LEVEL-A where communication is not guided by prior knowledge (see Figure 5.1). Specifically, this system includes the following parts:

- An information source that produces data to be transmitted to the destination. The information is implied by various modalities (e.g., text, audio, image, video, and their combinations). Data of different modalities are characterized by different levels of information density and redundancy.

- A local knowledge base that provides additional guidance for the semantic transceiver. This background knowledge base exists in various formats:

  - Knowledge graph;
  - Database;
  - Trained parametric or non-parametric models;

  It also provides a priori knowledge about the channel state information for both transmitter and receiver. With the knowledge base, the semantic transmitter achieves effective signal processing, which stresses the important parts of source data and matches the characteristic of the channel as well.

- A semantic transmitter that operates on the source data to extract the semantics-related features, which can be categorized into multiple semantic streams, each corresponding to one type of semantic feature of source data. In addition, each semantic feature corresponds to different semantic importance for specific tasks at the receiver, including data recovery or downstream intelligent tasks, such as language translation and object detection. The features are further processed and unequally protected according to the channel states.

- The technical transmitter, physical channel, and technical receiver operate on the semantic feature sequence of each semantic stream in some way to produce signals suitable for transmission over the physical channel. It can utilize the traditional separation-based source and channel coding combined with modulation to produce digital signals. It can also directly generate analog signals by using the emerging joint source channel coding with deep neural networks. The technical receiver ordinarily performs the inverse operation of that done by the technical transmitter, reconstructing the message from the signal.
- The semantic receiver performs the inverse operation of that done by the semantic transmitter, exploiting semantic fusion to reconstruct source data or directly execute downstream intelligent tasks.

The semantic communication system incorporates a semantic transmitter that retrieves the source semantic feature, guided by the background knowledge and the specific task intended for execution at the receiver. Each compressed sequence X is linked to a set of S sequences that are jointly typical with the provided X. In this context, all the source sequences within this fan signify identical semantics. Although, in the traditional source coding perspective, this merging of source typical sequences implies lossy compression, the incorporation of background knowledge transforms it into a meaning-lossless process. This ensures the perfect transmission of semantics or the complete execution of the task. Consequently, the number of sequences within the compact semantic space is evidently less than that in the classical compressed signal space.

Task-oriented Communication and Split Computing: SOTA Approaches and Applications

Task-oriented communication is a new paradigm that aims at providing efficient connectivity for accomplishing intelligent tasks rather than reception of every transmitted bit. Liu et al. [160] propose a task-oriented communication architecture tailored for end-to-end semantic transmission. They utilize the adaptable semantic compression (ASC) method to compress extracted semantics. Addressing scenarios with multiple users in a delay-intolerant system poses a challenge. The authors emphasize the importance of balancing compression ratios since higher ratios conserve channel resources but may induce semantic distortion, while lower ratios demand more resources and risk transmission failure due to delay constraints. To address this challenge, authors optimize both compression ratio and resource allocation to maximize the probability of task success.

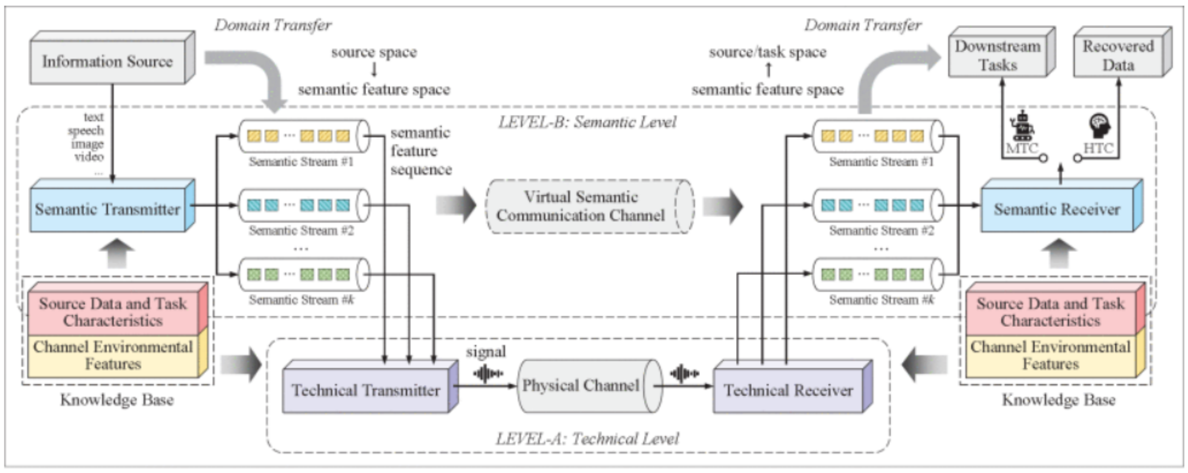


Figure 5.1: Block diagram illustrating a semantic communication system designed to facilitate both human-type communications (HTC) - LEVEL-B - and machine-type communications (MTC) - LEVEL-A.

For text files, Kutay et al. [148] introduce semantic quantization and compression methods for text, employing sentence embeddings and a semantic distortion metric to maintain meaning. The results reveal significant savings (orders of magnitude) in the necessary number of bits for message representation, with only minor accuracy loss compared to the semantic agnostic baseline. The study compares the outcomes of the proposed approaches, indicating that resource savings achieved through semantic quantization can be enhanced by incorporating semantic clustering. Notably, the methodology exhibits generalizability, delivering excellent results across various benchmark text classification datasets with diverse contexts.

While intelligent computer vision tasks remain a pivotal use case, the applicability of Edge Intelligence faces limitations due to communication-related constraints. A promising insight emerges, suggesting new approaches. Conventional image compression techniques primarily cater to human perception rather than facilitating image analysis. Consequently, achieving high performance requires transferring substantial data volumes over channels with limited capacity. To address this issue, *SC* (also referred to as supervised compression in some contexts) [113, 196] have recently emerged as a promising and viable alternative that achieves state-of-the-art performance in computer vision tasks while effectively reducing bandwidth usage. The idea is to incorporate encoder/decoder-like structures within the ML models themselves, and use specialized training techniques to train task-oriented compressed representations [253, 254]. Knowledge Distillation [170, 171] is one of the tools used to maximize the effectiveness of *SC* frameworks, where the altered model representation is trained to mimic the behavior of the original model.

Cunico et al. [62] present a significant advancement in split computing, addressing the challenge of determining optimal points for dividing a deep neural network to accommodate part of it on an embedded device and the remainder on a server. Unlike prior methods that rely solely on architectural aspects, such as layer sizes, to identify potential split locations, their approach considers the importance of individual neurons within the layers. A neuron is deemed important if its gradient concerning the correct class decision is high. The proposed Interpretable Split (I-SPLIT) procedure identifies suitable splitting points by predicting their performance in terms of classification accuracy before actual implementation. Notably, I-SPLIT demonstrates that the optimal splitting point in multiclass categorization depends on the specific classes the network addresses. The effectiveness of I-SPLIT is validated through extensive experiments on VGG16 and ResNet-50 networks across three datasets: Tiny-Imagenet-200, notMNIST, and Chest X-Ray Pneumonia.

5.2 Furcifer: Container Based Context-Adaptation for Object Detection

Modern real-time applications frequently integrate compute-intensive neural algorithms. Existing solutions either deploy highly-optimized Deep Neural Networks (DNNs) on mobile devices, leading to higher energy consumption and lower performance, or offload the execution of larger, higher-performance neural models to edge servers, requiring low-latency wireless data transfer. The optimality of these configurations, in terms of energy consumption, task performance, and latency, depends on time-varying variables like connection quality and system load. This paper introduces Furcifer, a framework dynamically adapting the computing configuration based on the perceived system state. Utilizing a container-based approach with low-complexity predictors, Furcifer demonstrates generalizability across operating environments. The framework also features a highly optimized split DNN model that achieves in-model supervised compression and enhances task offloading. Experimental results for Object Detection under diverse conditions, environments, and wireless technologies highlight Furcifer's significant benefits, including a 2x reduction in energy consumption, 30% higher mean Average Precision score than pure local computing, and a notable three-fold increase in frame per second rate compared to static offloading.

5.2.1 Computing Strategies Comparison:

The most popular metric used to evaluate OD is mean Average Precision (*mAP*), which combines precision and recall values based on Intersection over Union (IoU) scores across

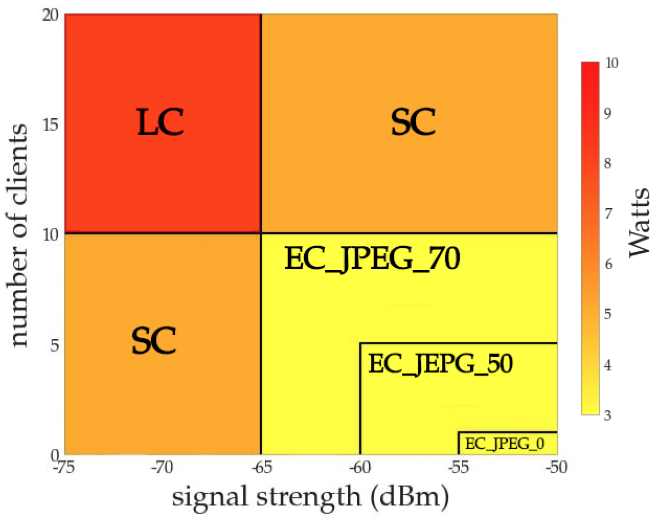


Figure 5.2: Best performing computing modality and associated MD's power consumption as a function of signal strength and number of connected users.

various levels of confidence thresholds. Typically, *mAP* scores are obtained by testing the algorithm on benchmark datasets such as COCO2017. However, when deploying an OD engine in a real-world setting, various factors such as camera resolution or scaling factor alterations come into play to determine the performance perceived by the application. Additional factors such as model quantization and image compression also play a significant role. With these in mind, we conduct an extensive evaluation of *EC* and *LC*, as well as Furcifer's *SC* engine. As *MD*, we select a Jetson Nano Dev Kit device, connected to a Jetson AGX Orin Dev Kit that acts as *ES*. We choose this specific family of microcontrollers as *MD* and *ES* for our tests for its efficient GPU hardware accelerated computing capabilities. To connect the MDs and *ES* we use IEEE 802.11n [266] and IEEE 802.11ac Wi-Fi connections. At the *ES*, we deploy a modified version of Faster R-CNN [217] with Res50 backbone as a feature extractor testing various JPEG compression rates: 0%, 50%, and 70%. We also explore high-frame-rate alternatives for *LC*. Specifically, we select a quantized FP16 version of YOLOv5 [136] and SSD300, a customized adaptation of the Single Shot MultiBox Detector (SSD) [163] developed by NVIDIA.

In our exploration of *SC*, we develop a specialized encoder-decoder architecture trained using supervised compression and Faster R-CNN as a teacher model. Our design is based on the model proposed in [174]. However, we optimized the original model by quantizing the encoder to FP16 and running it with an optimized inference engine. In addition, we fine-tuned the student model in order to match the camera resolution with the feature extractor upscaling factor.

OD Computing Configuration	mAP	FPS± std	FPS$_{min}$	FPS$_{max}$
EC_JPEG_0	37.051	3.29 ± 1.264	1.2	5.52
EC_JPEG_50	31.797	6.46 ± 3.007	1.89	10.01
EC_JPEG_70	29.476	6.66 ± 3.058	2.01	10.32
SC_FURCIFER	25.964	8.38 ± 1.992	4.96	11.72
LC_YOLOv5(FP_{16})	23.403	13.46 ± 0.261	11.89	13.89
LC_SSD$_{300}$(FP_{16})	23.201	28.45 ± 0.635	27.83	33.12

Table 5.1: Mean Average Precision and FPS statistics depending on OD computing configuration

The data reported in Figure 5.2 and Table 5.1 provides a comparison of the frame per second (FPS) and *mAP* obtained by each computing modality and model, whereas the figure shows the computing modality achieving the best FPS rate and the associated power consumption (MD only) as a function of signal strength (MD to ES channel) and the number of connected users. The results show that the best *mAP* performance is obtained using *EC* without JPEG compression - that is, the largest model running without image compression. Conversely, the maximum frame rate is achieved by a quantized version of the original model deployed on the *MD*. The figure illustrates how the optimal configuration is a function of the state of the channel and how different options result in a different amount of power spent by the mobile device. It should be pointed that this refers to the power consumed by *MD* and it does not take into account the overall total energy consumed by the whole system. As demonstrated later, Furcifer outperforms *EC* in terms of speed by achieving up to twice the FPS rate, all while achieving a higher *mAP* score in low-channel quality scenarios compared to the *LC* models.

5.2.2 System Design and Architecture:

The quantitative indicators presented in the previous section emphasize how there is no absolute winner among *EC*, *SC*, and *LC* even when considering a specific task, computing platforms, and communication technology. Instead, the – time varying – state of the system, which is influenced by mobility and load dynamics, determines the best computing configuration. However, changing the computing modality in real-world deployments is technically non-trivial. Furcifer realizes an adaptation engine composed of highly effective containerized models whose activation is determined by a control module informed by comprehensive system monitoring. While every element within the system holds a crucial role to enable adaptation to context, the container-based Service-Oriented Architecture (SOA) nature of Furcifer enables the independent deployment of each component. Furcifer tackles this obstacle by encapsulating each computing strategy within a container. Such containerization bundles the code and its dependencies, ensuring the ML application runs swiftly and reliably across diverse computing environments and facilitating a seamless transition between running ML models based on specific context characteristics. We have implemented Furcifer approach to Object Detection (OD) models as an ML task that is representative of real-time applications. In contrast with the majority of existing academic frameworks, Furcifer integrates optimized low-level operations in a transparent and easy-to-use manner by providing well-tuned high-level interfaces to the application layer. Within this section, we provide an in-depth discussion of the main features of each component while graphically representing the overall architecture and component unit in Figure

From the Cloud the Edge: On-Demand Image Pulling

Before Furcifer, the adoption of containerized models on mobile devices was limited by the lack of hardware acceleration support and excessive computational power required. Our middleware integrates GPU-enabled capabilities offered by the original Docker runtime in a lightweight version of the renowned container framework where unused modules were removed. Furcifer provides a specialized container registry specifically designed for image compression and storage. This registry stores well-tailored images optimized for each compatible device, which are cached for future use based on the specific task the device is assigned. For each device type, a subset of images shares identical interfaces with the operating system hypervisor. However, these images differ at the application layer and user library level, adapting to the specific task to be executed and the corresponding dependencies that are necessary.

This approach ensures minimal network usage, as only the final layer of the container varies from the previous one, while the operating system platform and essential libraries remain consistent. Notably, it is quite reasonable to expect that the operating system and the majority of user libraries will remain unchanged when deploying a new model to address an incoming task. By adopting this approach, the well-established concepts utilized in cloud environments streamline the management and real-time adaptation of mobile and resource-constrained devices. This shift of paradigm from the cloud to the edge empowers proactive mechanisms that enable seamless adjustments in response to evolving context requirements. We choose to apply containerization as a practical way to guarantee flexibility and fast reactiveness of the framework to future environment states. Our evaluation of the size of resulting container images reveals that less than 1% of the image comprises application-level files. This efficient design enables the download of only the last layer of the image, which is equivalent in size to the model itself. This approach minimizes network usage, since the model would anyway need to be transferred no matter whether a containerized approach is used or not.

Figure 5.3 shows the memory footprint of Furcifer container images in the *LC*, *SC* and *EC* configurations. Notably, the memory requirements for hardware acceleration dependencies

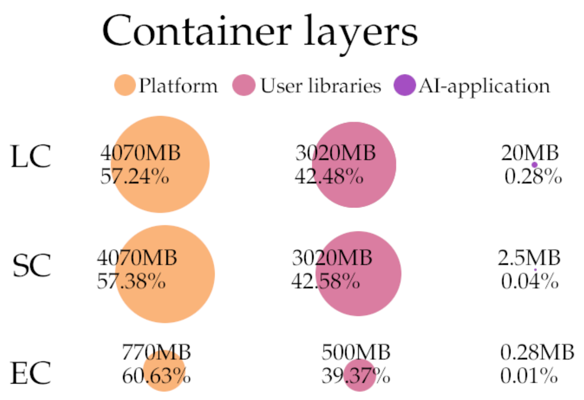


Figure 5.3: Furcifer: Components on *MD*.

at both the platform and user library levels are about four times more extensive for *LC* and *SC* compared to *EC*. Remarkably, the contribution of the application layer remains minimal compared to the other components, reinforcing the viability of switching between containers on demand. This is achieved by pulling only the necessary components, specifically the last layer of the container image, thereby reducing unnecessary network usage and resource consumption.

Communication Interface and Protocol:

Deployed containerized models corresponds to a distinct and uniquely identified end point on the *MD*. Those microservices interact with the central orchestrator through a REST API, facilitating seamless communication and interaction while capitalizing on the advantages of minimal communication overhead. It continuously monitors potential new connections and, in the event of a connection loss, takes informed countermeasures to address the situation by switching to a local computing strategy.

Furcifer's REST APIs operate on a request-response TCP-based model, enabling the framework to discern round-trip packet latencies. This functionality allows the introduction of well-defined rules for filtering out requests characterized by excessive communication delays. This mechanism ensures that the framework remains responsive and efficient, even in scenarios where the network conditions might fluctuate. Exchanged messages are defined as follows:

- **keep_alive**: This message is sent periodically using a polling mechanism to ascertain the presence of mobile devices within the same network.

- **start/stop_OD**: This message instructs the Mobile Device *MD* to initiate or terminate an Object Detection task. When initiating a task, the message also specifies the preferred computing strategy among *LC*, *SC*, and *EC*.

- **release_camera**: Since the camera is a shared resource among deployed models on the *MD*, this message prompts the *MD* to release the camera lock, enabling other models to access the camera.
- **set_target_frame_rate**: This command sets the desired *FPS* rate for camera sampling based on dynamic requirements defined at the application level. Recognizing the direct correlation between higher *FPS* rates and increased power consumption, Furcifer intelligently conserves energy and network resources when higher frequency camera sampling is unnecessary, e.g., Vehicle-to-Vehicle (V2V) cameras in low-traffic environments [250]).
- **set_target_resolution**: Recognizing that higher image resolution corresponds to increased complexity in convolutional operations, extended processing time, and elevated energy consumption, Furcifer offers the flexibility of dynamically defining camera resolutions. This empowers the system to tailor image resolution to the specific demands of the prevailing scenarios, ensuring a judicious trade-off between computational efficiency and detection fidelity.
- **set_compression_rate**: If an *EC* configuration is used, the *MD* can opt to compress captured images before transmitting them to the *ES* for final detection. This message specifies the desired compression rate, controlling the balance between reduced compression for improved *mAP* score.
- **get_metrics**: This request explicitly prompts the *MD* to share its current performance metrics. Those encompass the active model, the current *FPS* rate, instantaneous power consumption, and the number of dropped frames due to excessive processing delays.

Furcifer's Semantic Transmitter

Furcifer introduces a new *SC* engine tailored for resource-constrained devices, marking a significant advancement in this real-world domain. Leveraging Faster R-CNN as the teacher model, we use a modified version of the knowledge distillation process adopted in SC2 Benchmark [174] to design a compact encoder optimized for constrained devices. This encoder serves a dual purpose: minimizing channel occupancy and effectively distributing computation load between mobile devices and the edge server. Differently from the original model described in [174] we optimized each tensor operation to exploit the parallel execution on the GPU. We transform the canonical matrix operations into cuDNN tensor operations [55] and run the model by building a TensorRT engine [55] for high performance inference optimization. In the fine-tuning process, we employ preprocessing image upscaling transformation of 400 pixels, which is smaller than that of the original model, with a minimum height of 800 pixels. This resolution adjustment is implemented to facilitate quicker inference on resource-constrained devices.

The optimized encoder heavily relies on quantization and channel compression to reduce execution time as much as possible. To enhance data compression, we strategically place a one-channel bottleneck in the initial layers of the feature extraction segment of the network. This choice leads to further data reduction, increasing the efficiency of the whole process. Additionally, we incorporate INT8 quantization at the end of the encoder. This quantization approach optimizes the representation of the data, contributing to both improved data compression and streamlined computation. The dynamic nature of the system is upheld by calculating the scaling factor and zero point on a per-image basis as they are processed. These values are then communicated to the decoder located at the *ES*, along with the resulting INT8 tensor from the encoder inference process.

The incorporation of INT8 quantization and dequantization for the inference tensor on the *MD* minimizes the influence of weight quantization within the encoder engine, which transitions from FP32 to FP16. We confirm the negligible impact on *mAP* score testing Furcifer's *SC* engine on the COCO2017 dataset, obtaining 25.966 and 25.964 as *mAP* scores for FP32 and FP16, respectively. Thus, the adoption of an FP16 quantized encoder on the mobile device delivers a nearly twofold increase in processing speed compared to its FP32 counterpart, while preserving about the same *mAP* score. This finding underscores the additional advantage of applying quantization to *SC* encoders, which, unlike their *LC* counterparts, are already quantizing the final encoding result to minimize channel occupancy. As a result, they are less susceptible to *mAP* score reduction due to quantization. We remark how this optimization makes *SC* a competitive option against optimized models for embedded devices, as demonstrated by our results.

Pareidolia: Low-Complexity Similarity-Based Context Adaptation

Pareidolia, a concept rooted in human perception, reflects the inclination to perceive distinct, often meaningful shapes or images within random or ambiguous visual patterns. It manifests as a natural cognitive process, wherein the brain attempts to link novel ideas with existing concepts. Leveraging already solved tasks, Furcifer leverages "pareidolia" as a context adaptation approach. Each participating *MD* maintains a record of previously completed tasks. This historical context empowers the node to discern which computing strategy aligns best with the current system state by identifying analogous past scenarios. When a sufficiently similar context is detected, the *ES* intervention may not be required. Conversely, if an analogous context is not found, pertinent task details are shared with the *ES* to collaboratively determine the optimal model and computing configuration (*EC*, *LC* or *SC*) that best matches the current system state. The Pareidolic Policy Manager (*PPM*), as defined, facilitates low-complexity trend forecasting [125, 169]. It has demonstrated its effectiveness across a diverse range of real-world scenarios, operating seamlessly without simplifications while reducing the minimal additional burden on already limited computational capabilities. This predictive process is specifically suited for constrained devices, requiring minimal additional burden on their already limited computational capabilities. As a result of the interaction between the increasing number of concurrent clients and the variability of network conditions, *PPM* forecasts the expected number of FPS that the mobile device will achieve when employing each considered computing configuration. This predictive functionality enables *PPM* to anticipate the impact of each strategy choice on FPS and tailor the decision accordingly. *PPM* forecasting capability comes from a set of predictors which based on the current metrics collected by Energon are capable of determine the resulting FPS rate the framework will achieve as a consequence of choosing a specific computing configuration. Additional insights into this module and its performance comparison against a more complex Deep Reinforcement Learning agent are discussed in Section.

5.2.3 Experimental Evaluation over Different Networking Conditions:

In this section, we present and report the result of the experiments carried out using the Furcifer framework. These experiments report relevant performance metrics on a broad range of states and settings of the targeted deployment environment, including outdoor and indoor ones covered with IEEE 802.11n and 802.11ac connectivity. Through this extensive set of experiments, we aim to assess the ability of Furcifer to dynamically adapt the cloud continuum configuration against system state dynamics. The dataset we collect comprises over 250 distinct combinations of channel conditions (expressed as signal strength) and number of concurrent client connections. Although indoor and outdoor experiments exhibit similar trends, indoor scenarios tend to be characterized by a higher degree of unpredictability, which is primarily due to the presence of obstacles that complicate signal propagation. As a result, the overall channel quality is adversely affected, leading to more variable and less consistent performance outcomes.

IEEE 802.11n Experiments:

First, we focus on the widely used Wi-Fi 801.11n standard. Our experimental setup features a Jetson Nano DevKit as the mobile device equipped with a 640x480px USB webcam. In indoor scenarios, we orchestrated the movement of the device along a designated path spanning approximately 20 meters. In outdoor scenarios, the path extends over a distance of 50 meters. This deliberate variation allowed us to replicate a spectrum of signal strengths and network dynamics, capturing the intricacies of both indoor environments and outdoor settings. For each experimental run, we meticulously examine the system scalability across different user scenarios. Specifically, we investigate the performance of the system, as the number of clients varies between 1 and 20.

Figures 5.4a and 5.4b depict the FPS and error percentage metrics for *SC* and *EC* with JPEG compression gain 0, 50 and 70% as a function of distance with a single connected user. Within the indoor experiment settings (Figure 5.4a), there is a striking similarity in the average FPS achieved by *EC* and *SC*, with the exception of the scenario involving no image compression. Conversely, in the outdoor environment (Figure 5.4b), *SC* takes advantage of the improved channel conditions compared to the indoor setting and achieves up to 2 additional FPS. Importantly, it is worth noting that *SC* not only excels in FPS but also achieves a higher *mAP* score compared to *LC*, showcasing its suitability in terms of both task performance and frame processing speed.

Figures 5.4c and 5.4d show performance metrics as the number of clients connected to *ES* varies. Several notable trends emerge from this analysis. First, we observe an interesting pattern regarding the impact of image compression techniques as the number of concurrent clients increases. The failure rates associated with JPEG are dramatically larger compared to *SC*, reaching up to 100% failure rate in some measurements. This result underscores the perils of relying solely on image compression when dealing with a larger number of clients, highlighting the potential limitations of this approach in dynamic and demanding network/server conditions. On the contrary, *SC* achieves almost a steady 0% failure rate due to the small amount of network and server resources used by this configuration. The resulting improved resilience of Furcifer underlines the effectiveness of its context-aware approach, which enables *SC* to consistently outperform *EC* in the range of tested network conditions.

IEEE 802.11ac Experiments:

The overhead in a wireless communication channel fluctuates based on the technology employed, thereby influencing the trade-off between computing and wireless communication inherent in the *SC* and *EC* conditions. We extend the evaluation of Furcifer performance to include IEEE 802.11ac. In these experiments, we replicated the same path for the MD, while concurrently running increasing parallel connections of up to thirty clients. We use a Wi-Fi network interface that supports the IEEE 802.11ac 5GHz protocol and establish a connection between the *ES* and *MD* over an 80 MHz band. Our Wi-Fi 5Ghz antennas also support MU-MIMO technology which additionally improves the overall communication performance.

Figure 5.5 shows FPS rate as a function of the number of concurrent clients, ranging from 10 to 30. All *EC* compression strategies benefit from improved connection capabilities, outperforming *SC* when the number of clients is smaller than 10. Instead, when the number of clients increases to 20 and 30, the additional load on *ES* penalizes *EC* over *SC* of about 40% in terms of the average FPS successfully processed over time. As shown in the analysis that follows, the superior performance compared to *ES* when the system is under pressure is due both to the decrease in channel usage and the effort of the server granted by *SC*. On the contrary, when the full capacity of the network and server are available, *SC* is penalized by the computing effort allocated to *MD*.

To better appreciate the impact of communications on the computing mode, we conduct a comparative analysis between the IEEE 802.11n and 802.11ac Wi-Fi protocols. The results shown in Figure 5.6 depict the FPS achieved as the number of connected clients increases. We note how the improved data rates offered by the IEEE 802.11ac means that *EC* is the winning solution up to a certain load level, whereas in IEEE 802.11n experiments, the superior compression granted by *SC* results in the latter being the winning solution. However, this

Table 5.2: Low complexity frame rate predictors

	indoor		outdoor		indoor $\rightarrow$ *outdoor*	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
EC_J0	0.27	7.30	0.37	10.69	0.58	20.37
EC_J50	0.50	7.86	0.59	9.25	0.71	15.37
EC_J70	0.47	7.48	0.53	7.68	0.81	15.98
SC	0.70	7.46	0.97	9.48	1.23	15.63

comparison is made with the same signal strength variability for *EC* and *SC*, moving the *MD* along the same spatial trajectory.

5.2.4 PPM: Pareidolic Policy Management

In the previous sections, we demonstrated the need for the dynamic adaptation of the computing configuration. We now evaluate the ability of Furcifer - and specifically its policy management module - to provide adaptation capabilities without imposing a significant overhead. In terms of energy consumption and optimal task performance, computing configurations can be clearly ranked based on the *MD* perspective. In fact, *EC* does not impose any computing load to the *MD* and achieves the best mAP thanks to the use of larger models. *SC* allocates minimal computing effort to the *MD* and has the second best mAP. Finally, *LC* results in the largest energy intake and worst performance. Thus, the decision engine has to evaluate the ability of the individual computing configurations to achieve the desired FPS rate given the currently perceived system state, and then select them in the order dictated by energy and mAP. To support such decision process, we then build simple KNN regressors that take as input application context metrics sush as: the inference time on the *ES*, the average round trip time, the communication channel quality, and the current resource usage. This produces as output the predicted FPS rate for each considered computing configuration, allowing *PPM* to anticipate the resulting Quality of Service (QoS) of a particular action before executing it. We evaluate the regressors when training and applying them in specific environments (e.g., indoor or outdoor), as well as on their ability to generalize. The resulting loss metrics are reported in Table 5.2. It can be observed that when these regressors are trained in the same context where they are applied, the error is minimal (below 11% *MAPE* – Mean Absolute Percentage Error) across all computing configurations. In the case of transfer learning, where models trained indoors are deployed outdoors, the maximum loss value *MAPE* increases to 20%.

Table 5.3: RMSE between percentage of OD computing configurations

	target_FPS	*PPM*	DRL
	4	7.848	87.210
	5	2.287	104.979
	6	21.875	123.547
	7	32.552	101.824
	8	36.892	75.419
	9	7.465	78.375
	10	24.045	69.928
	11	58.593	71.310
$\overline{RMSE}$		23.944	89.074

Adaptation Results:

We compare Furcifer performance with a static *LC* solution, which represents the only viable option when the connection quality or system load cannot support the desired FPS rate. Figure 5.7 shows the distribution of decisions made by Furcifer policy manager across the spectrum of available cloud continuum strategies. It is important to note that our policy manager, despite its low complexity, demonstrates the ability to match the configuration that an oracle controller would implement. Table 5.3 reports the Root Mean Squared Error (RMSE) between the percentage of choices made by Furcifer *PPM* low complexity policy manager and a baseline DRL agent. The striking alignment between the decisions made by the Furcifer pareidolic policy manager and the ground truth highlights the feasibility of deploying a low complexity predictor deployed at *MD*, where more complex controllers may fail to train properly or adequately generalize.

In the IEEE 802.11n configuration, Furcifer reduces the energy intake by approximately 80% while achieving an average *mAP* score increase of over 20% in comparison to *LC*. The relevance of these outcomes is further amplified when using the IEEE 802.11ac protocol. In this scenario, energy savings exceed 100%, and *mAP* consistently maintains a level above 20% for all defined targets FPS. As Figure 5.8 shows, Furcifer can easily generalize, accurately predicting the frame per second even when trained on an indoor environment and then tested outdoor.

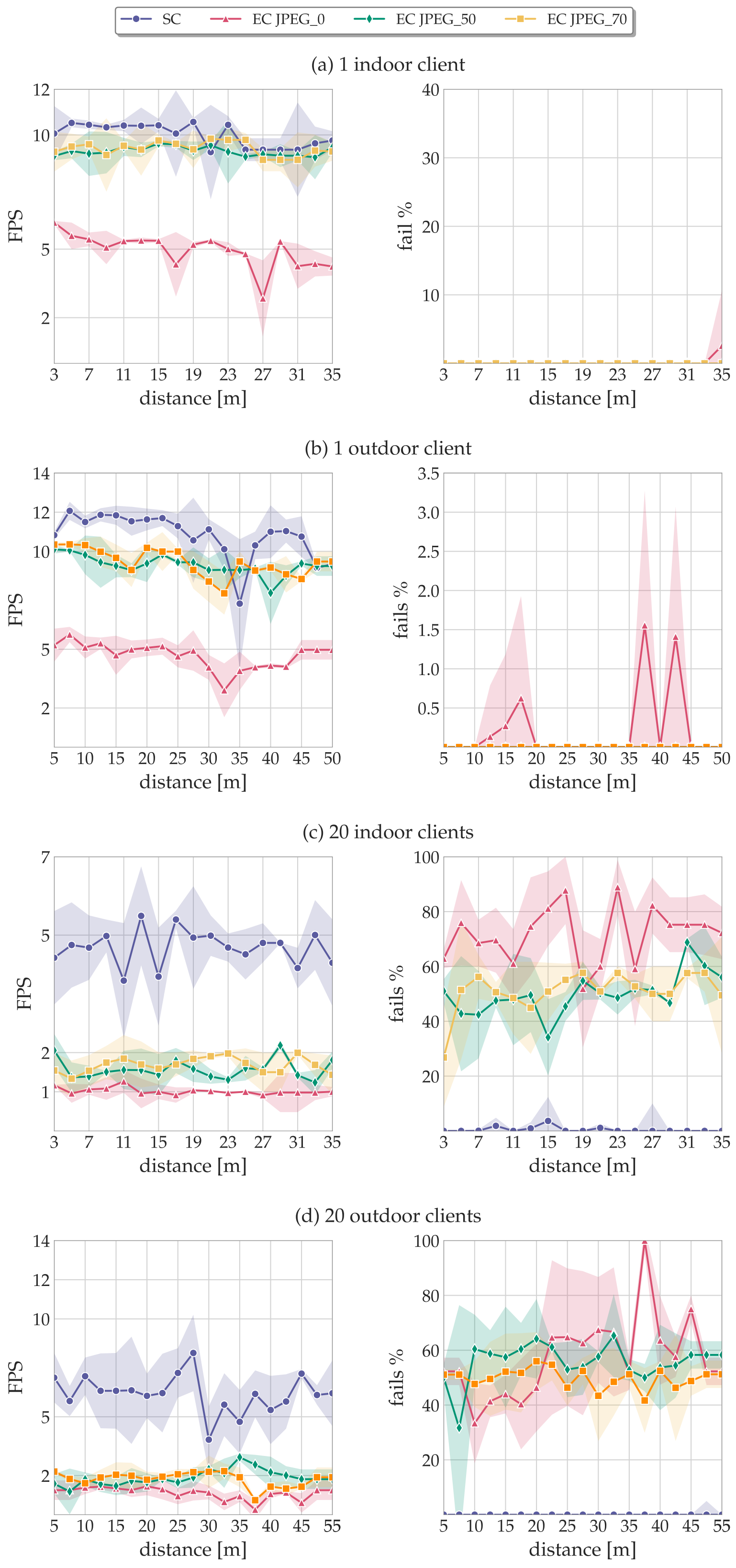


Figure 5.4: IEEE 802.11n experiments: with one connected client (a) indoors and (b) outdoors, and twenty connected clients (c) indoors and (d) outdoors. Performance metrics - FPS, fail percentage - as a function of distance for *SC* and *EC* with 0%, 50%, and 70% JPEG compression gains. The lines in the plots correspond to average metrics, while the shading represents variance.

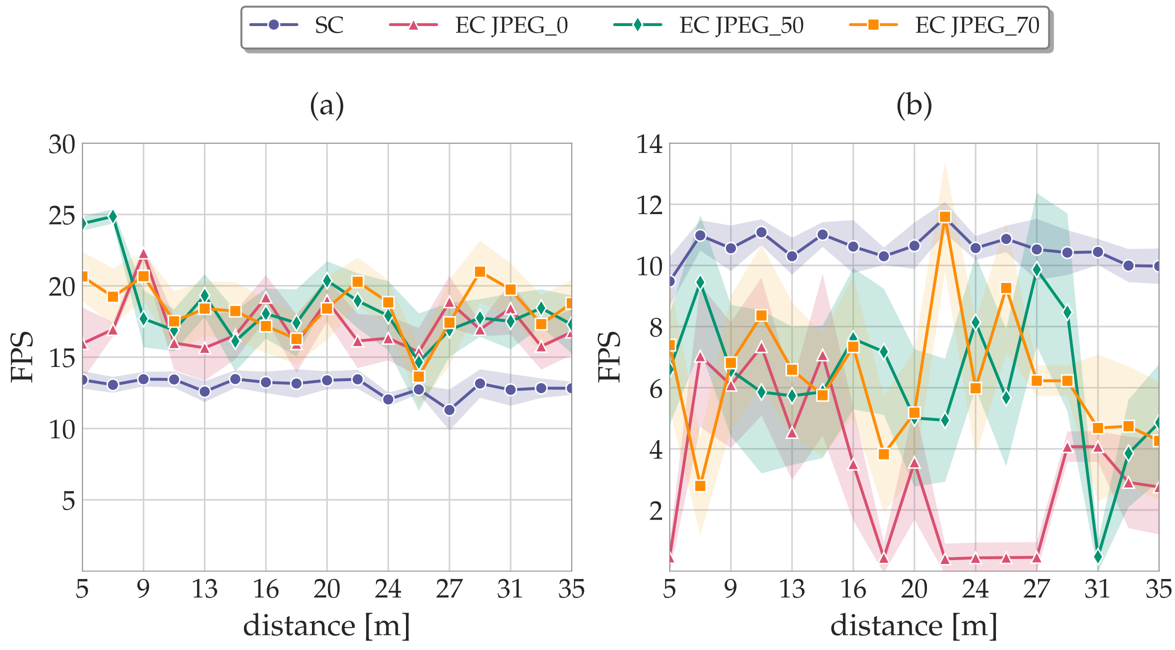


Figure 5.5: IEEE 802.11ac *FPS* rate metric: ten connected client (a) and twenty connected client (b) - as a function of distance for *SC* and *EC* with JPEG compression gain 0, 50 and 70%. . The lines in the plots correspond to average metrics, while the shade is the variance.

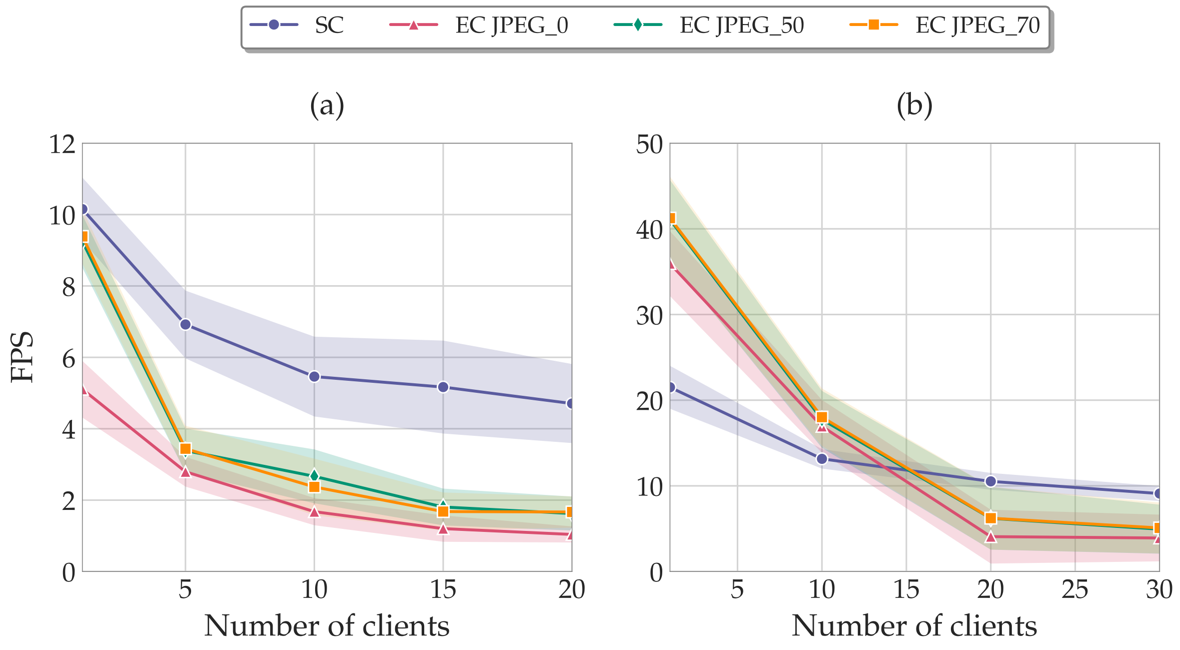


Figure 5.6: Comparison between the 802.11n (a) and 802.11ac (b) Wi-Fi protocols, depicting the FPS_fails score with an increasing number of concurrent clients.

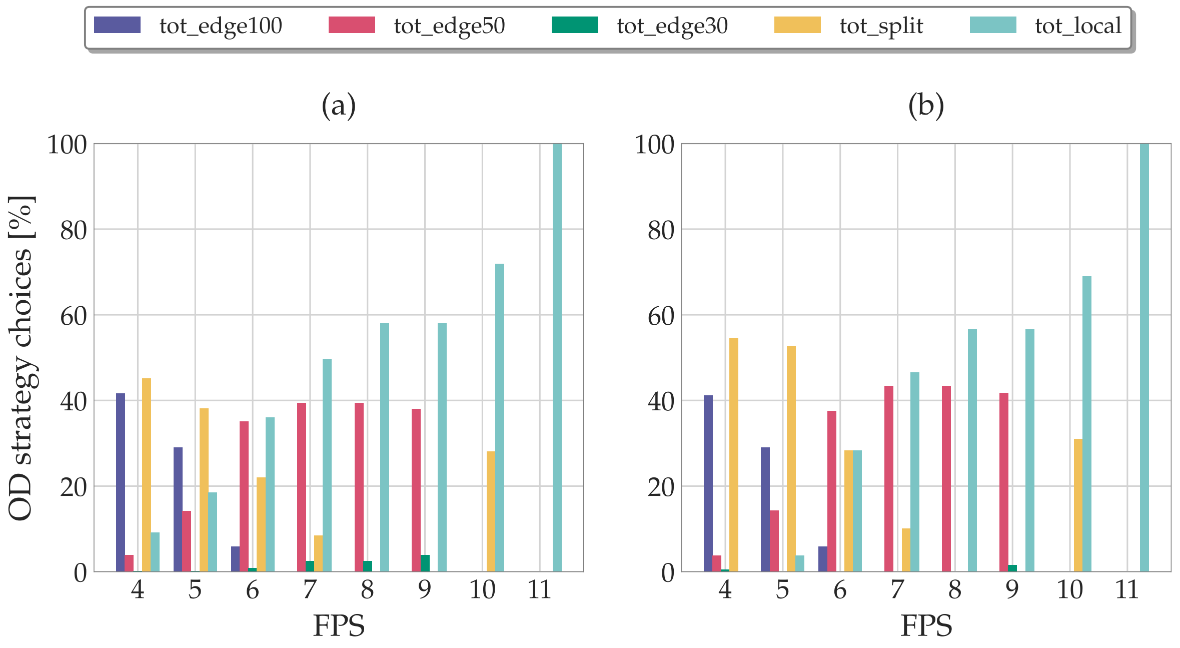


Figure 5.7: Choices distribution of considered computing configurations depending on target *FPS* rate for the Ground truth oracle (a) and our Pareidolic Policy manager (b)

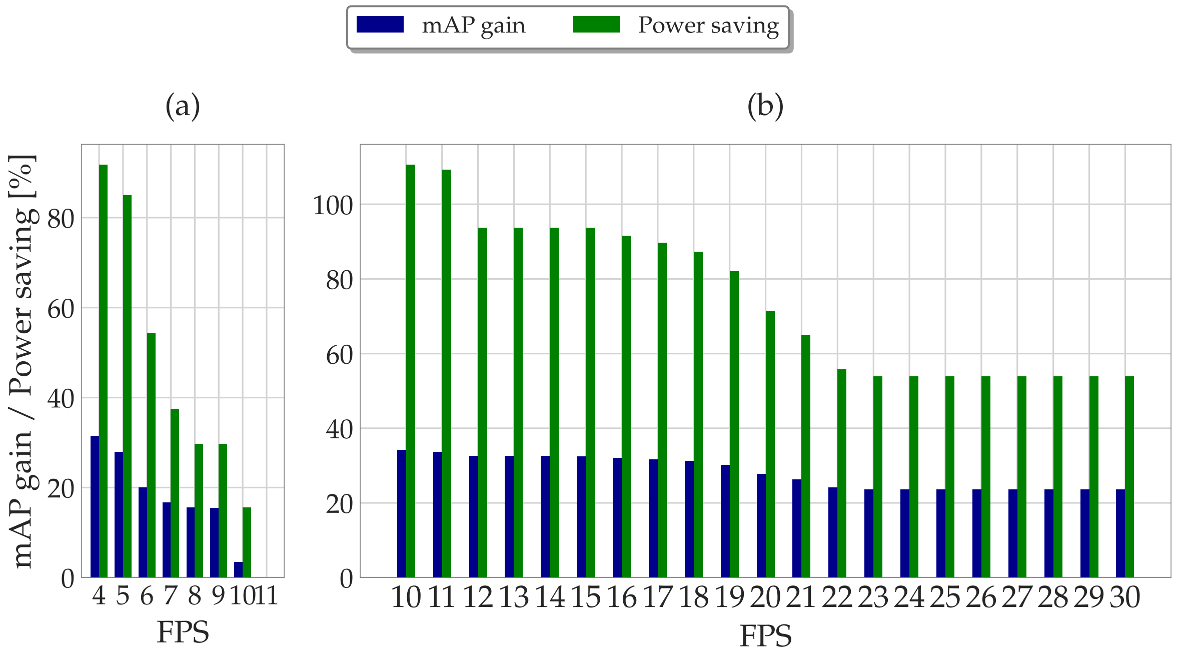


Figure 5.8: *mAP* gain and Energy saving with IEEE (a) 802.11n and (b) 802.11ac Wi-Fi protocols

Concluding Remarks

This thesis has navigated the dynamic landscape of computing, particularly at the intersection of Artificial Intelligence (AI), frugality, and serverless paradigms. Across different sections, transparent frugality emerged as a guiding philosophy, aiming for optimal efficiency in resource utilization without compromising the richness of AI functionalities. We highlighted the importance of addressing power efficiency in energy-constrained environments, proposing greener and more sustainable approaches in distributed learning. The research has addressed and provided solutions to the challenges associated with resource constraints in the deployment of intelligent agents on constrained devices.

Beginning with cloud-native solutions, we have embraced various state-of-the-art (SOTA) approaches for the transparent management of networking resources. Within this journey, digital twins (DTs) [24] have served as a reference point, offering a foundation to achieve transparency and efficiency in heterogeneous Industrial Internet of Things (IIoT) environments. An impactful experimental contribution is showcased in edge and industrial scenarios [26], validating the practical application of digital twin technology in handlig up to 1000msg/sec with an average overall delay of 6ms. This illustration highlights the potential for attaining dynamic and transparent management of communication resources in IoT environments. Furthermore, a novel smart routing approach is introduced in [27], exemplifying the advantageous integration of AI in computing tasks by routing incoming packets in less than 250ms while avoiding packet dropping.

The initial constraint scrutinized in this thesis is power consumption. Skillfully navigating the delicate balance between performance and energy trade-offs, we delve into the critical aspects of energy monitoring and optimization within edge-constrained and intelligent environments. Moreover, the research introduces two environmentally conscious approaches to distributed learning [10, 180], specifically considering power consumption in the planning strategies for Federated Learning rounds. This comprehensive exploration aims to contribute to the development of greener and frugality based practices in the realm of distributed learning,

fostering efficiency and eco-friendly strategies in power-aware intelligent systems. Achieved results include up to 60% of energy savings and 3.5 times faster convergence against popular client selection approaches.
Continuing our exploration, we turn our attention to challenges associated with data-constrained scenarios. In this context, we confront privacy and trust-related issues within a Federated Learning setting, presenting a novel contribution [175]. The resulting decentralized oracle networks (DON) exhibited approximately 10% enhanced accuracy across three distinct privacy-critical use cases. Moreover, acknowledging the scarcity of data and the intricacies of model complexity, we propose an ensemble of weak autoregressors as an innovative solution for traffic volume forecasting [181] with up to 98.64% prediction accuracy. This dual approach not only addresses the pressing issues of privacy and trust in federated learning but also offers a frugal yet effective strategy for forecasting in scenarios characterized by limited data and intricate model dynamics.

In our exploration at the nexus of AI, frugality, and serverless computing, we finally redirect our focus toward scenarios characterized by unstable and unreliable networking conditions. Addressing the offloading of AI-based tasks, we adopt a communication-aware approach. Embracing the Split Computing paradigm, we introduce a distilled encoder [183] that stands poised to rival the performance of deeper neural networks. This distilled encoder facilitates real-time semantic compression on mobile devices, thereby presenting a pragmatic solution tailored to the challenges posed by unpredictable networking environments. The experimental results underscore the superior performance of our middleware compared to both local and edge computing. In particular, Furcifer achieves outstanding outcomes, showcasing a notable 2x reduction in energy consumption and a 30% additional mAP score gain compared to local computing. Additionally, it achieves an impressive three-fold increase in the FPS rate compared to edge computing.

Acknowledging the inherent limitations of this thesis, which are based on specific preliminary experimental scenarios and assumptions, we argue that the comprehensive and unified perspective presented here on Frugality-based techniques represents a practical step toward embracing more sophisticated and intelligent systems in energy-, data-, and communication-constrained environments.

Looking ahead, given the additional computation brought by larger Machine Learning models, our future work will be guided by two main assumptions:

(I) As the pendulum swings towards decentralization from centralized IT infrastructure, there is a growing need for advanced and transparent optimization strategies. These strategies are essential to support more complex Artificial Intelligence models at the Edge.

(II) Simultaneously, the larger size of data collected on the Edge will require more advanced communication methods, especially in cases where local edge computing capabilities are insufficient to locally process incoming data. AI-enhanced semantic communication holds the promise of unlocking new possibilities, thereby pushing the boundaries of frugality in the realm of intelligent and distributed computing.

For these reasons, our first focus will be on exploring optimization strategies for AI on the Edge, aiming to minimize energy and data usage on constrained devices while prioritizing transparency and usability. This exploration will involve a detailed analysis of the implications of the "From the Cloud to the Edge" paradigm, assessing both its benefits and limitations, especially regarding containerization in resource-limited scenarios. Secondly, we will extend the task-aware compression method employed in Furcifer [183] from Object Detection to various non-vision distributed tasks, with a focus on enhancing robustness and efficiency in dynamically evolving environments.

Bibliography

[1] MlFlow: An open source platform for the machine learning lifecycle.

[2] PowerTOP: A Linux tool for power consumption management.

[3] Solar power plant time series analysis v5 01-22. kaggle.com/code/elbiopea/solar-power-plant-time-series-analysis-v5-01-22/. Last Accessed: 2022-09.

[4] A. Ziller *et al.* *PySyft: A Library for Easy Federated Learning*, pages 111–139. Springer International Publishing, 2021.

[5] Mohammad Aazam, Imran Khan, Aymen Abdullah Alsaffar, and Eui-Nam Huh. Cloud of things: Integrating internet of things and cloud computing and the issues involved. In *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014*, pages 414–419. IEEE, 2014.

[6] Mohamed Abdel-Basset, Nour Moustafa, and Hossam Hawash. Privacy-Preserved Cyberattack Detection in Industrial Edge of Things (IEoT): A Blockchain-Orchestrated Federated Learning Approach. *IEEE Transactions on Industrial Informatics*, 18(11):7920–7934, 2022.

[7] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. Cloud of things for sensing-as-a-service: Architecture, algorithms, and use case. *IEEE Internet of Things Journal*, 3(6):1099–1112, 2016.

[8] Joydeep Acharya and Sudhanshu Gaur. Edge compression of gps data for mobile iot. In *2017 IEEE Fog World Congress (FWC)*, pages 1–6. IEEE, 2017.

[9] Mainak Adhikari, Abhishek Hazra, Varun G Menon, Brijesh K Chaurasia, and Shahid Mumtaz. A roadmap of next-generation wireless technology for 6g-enabled vehicular networks. *IEEE Internet of Things Magazine*, 4(4):79–85, December 2021.

[10] Andrea Agiollo, Paolo Bellavista, Matteo Mendula, and Andrea Omicini. Enea-fl: Energy-aware orchestration for serverless federated learning. *Future Generation Computer Systems*, 2024.

[11] Faraz Ahmed, M. Zubair Shafiq, and Alex X. Liu. The internet is for porn: Measurement and analysis of online adult traffic. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 88–97, 2016.

[12] Shadi Al-Sarawi, Mohammed Anbar, Rosni Abdullah, and Ahmad B. Al Hawari. Internet of things market analysis forecasts, 2020–2030. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 449–453, 2020.

[13] Mojtaba Alizadeh and Wan Haslina Hassan. Challenges and opportunities of mobile cloud computing. In *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 660–666. ieee, 2013.

[14] R Ananthapadmanabha, BS Manoj, and C Siva Ram Murthy. Multi-hop cellular networks: the architecture and routing protocols. In *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2001. Proceedings (Cat. No. 01TH8598)*, volume 2, pages G–G. IEEE, 2001.

[15] Giacomo Angione, Paolo Bellavista, Antonio Corradi, and Eugenio Magistretti. A k-hop clustering protocol for dense mobile ad-hoc networks. In *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, pages 10–10. IEEE, 2006.

[16] Amna Arouj and Ahmed M. Abdelmoniem. Towards Energy-Aware Federated Learning on Battery-Powered Clients. *CoRR*, abs/2208.04505, 2022.

[17] Anil Aryal, Sangam Shrestha, and Mukand S. Babel. Quantifying the sources of uncertainty in an ensemble of hydrological climate-impact projections. *Theoretical and Applied Climatology*, 135(1–2):193–209, January 2018.

[18] Mohammad Babar, Muhammad Sohail Khan, Farman Ali, Muhammad Imran, and Muhammad Shoaib. Cloudlet computing: recent advances, taxonomy, and challenges. *IEEE access*, 9:29609–29622, 2021.

[19] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and*

Statistics, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.

[20] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting mobile 3g using wifi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys'10. ACM, June 2010.

[21] Yixin Bao, Yanghua Peng, and Chuan Wu. Deep learning-based job placement in distributed machine learning clusters. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 505–513, 2019.

[22] Sasan Barak and S. Saeedeh Sadegh. Forecasting energy consumption using ensemble arima–anfis hybrid algorithm. *International Journal of Electrical Power amp; Energy Systems*, 82:92–104, November 2016.

[23] David F Barbe. *Very large scale integration (VLSI): fundamentals and applications*, volume 5. Springer Science & Business Media, 2013.

[24] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671, 2019.

[25] Paolo Bellavista, Luca Foschini, and Alessio Mora. Decentralised learning in federated deployment environments: A system-level survey. *ACM Comput. Surv.*, 54(1), feb 2021.

[26] Paolo Bellavista, Carlo Giannelli, Marco Mamei, Matteo Mendula, and Marco Picone. Application-driven network-aware digital twin management in industrial edge environments. *IEEE Transactions on Industrial Informatics*, 17(11):7791–7801, 2021.

[27] Paolo Bellavista, Carlo Giannelli, Marco Mamei, Matteo Mendula, and Marco Picone. Digital twin oriented architecture for secure and qos aware intelligent communications in industrial environments. *Pervasive and Mobile Computing*, 85:101646, 2022.

[28] Paolo Bellavista, Carlo Giannelli, and Dmitrij David Padalino Montenero. A reference model and prototype implementation for sdn-based multi layer routing in fog environments. *IEEE Transactions on Network and Service Management*, 17(3):1460–1473, September 2020.

[29] Paolo Bellavista and Eugenio Magistretti. How node mobility affects k-hop cluster quality in mobile ad hoc networks: A quantitative evaluation. In *2008 IEEE Symposium on Computers and Communications*, pages 750–756. IEEE, 2008.

[30] Lucia Lo Bello, Alfio Lombardo, Sebastiano Milardo, Gaetano Patti, and Marco Reno. Experimental assessments and analysis of an sdn framework to integrate mobility management in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 16(8):5586–5595, August 2020.

[31] Djalel Benbouzid, Róbert Busa-Fekete, and Balázs Kégl. Fast classification using sparse decision dags. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1, 06 2012.

[32] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.

[33] Pravin Bhagwat. Bluetooth: technology for short-range wireless apps. *IEEE Internet computing*, 5(3):96–103, 2001.

[34] Neelanjan Bhowmik, Jack W. Barker, Yona Falinie A. Gaus, and Toby P. Breckon. Lost in compression: The impact of lossy image compression on variable size object detection within infrared imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 369–378, June 2022.

[35] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data, 2016.

[36] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[37] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gael Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In A. Smola, A. Dimakis, and I. Stoica, editors, *Proceedings of Machine Learning and Systems*, volume 3, pages 747–769, 2021.

[38] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1–12, 2018.

[39] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[40] Leo Breiman. *Machine Learning*, 45(1):5–32, 2001.

[41] Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, January 2021.

[42] Rajkumar Buyya, Saurabh Kumar Garg, and Rodrigo N Calheiros. Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *2011 international conference on cloud and service computing*, pages 1–10. IEEE, 2011.

[43] C Byers and R Swanson. Openfog consortium openfog reference architecture for fog computing. *OpenFog Consortium Archit. Working Group, Fremont, CA, USA, Tech. Rep. OPFRA001*, 20817:27–28, 2017.

[44] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[45] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping, 2022.

[46] Giuseppe Cardone, Luca Foschini, Paolo Bellavista, Antonio Corradi, Cristian Borcea, Manoop Talasila, and Reza Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, 2013.

[47] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. The rise of serverless computing. *Commun. ACM*, 62(12):44–54, nov 2019.

[48] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE transactions on services Computing*, 5(2):164–177, 2011.

[49] Yacine Chakhchoukh, Patrick Panciatici, and Pascal Bondon. Robust estimation of sarima models: Application to short-term load forecasting. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, pages 77–80, 2009.

[50] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, July 2009.

[51] Arunava Chatterjee, Subho Paul, and Biswarup Ganguly. Multi-objective energy management of a smart home in real time environment. *IEEE Transactions on Industry Applications*, 59(1):138–147, 2023.

[52] Aleksandra Checko, Henrik L Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S Berger, and Lars Dittmann. Cloud ran for mobile networks—a technology overview. *IEEE Communications surveys & tutorials*, 17(1):405–426, 2014.

[53] Hang Chen, Syed Ali Asif, Jihong Park, Chien-Chung Shen, and Mehdi Bennis. Robust blockchained federated learning with model validation and proof-of-stake inspired consensus. *arXiv preprint arXiv:2101.03300*, 2021.

[54] Yi-Lin Cheng, Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu. High resource utilization auto-scaling algorithms for heterogeneous container configurations. In *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 143–150, 2017.

[55] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning, 2014.

[56] Mung Chiang, Sangtae Ha, Fulvio Risso, Tao Zhang, and I Chih-Lin. Clarifying fog computing and networking: 10 questions and answers. *IEEE Communications Magazine*, 55(4):18–20, 2017.

[57] Hyomin Choi and Ivan V. Bajić. Deep feature compression for collaborative object detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3743–3747, 2018.

[58] Olga Chukhno, Nadezhda Chukhno, Giuseppe Araniti, Claudia Campolo, Antonio Iera, and Antonella Molinaro. Optimal placement of social digital twins in edge iot networks. *Sensors*, 20(21):6181, October 2020.

[59] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11. ACM, April 2011.

[60] Open Edge Computing. Open edge computing, 2018.

[61] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys'10. ACM, June 2010.

[62] Federico Cunico, Luigi Capogrosso, Francesco Setti, Damiano Carra, Franco Fummi, and Marco Cristani. I-split: Deep network interpretability for split computing. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2575–2581, 2022.

[63] Cheng Dai, Xingang Liu, Weiting Chen, and Chin-Feng Lai. A low-latency object detection algorithm for the edge devices of iov systems. *IEEE Transactions on Vehicular Technology*, 69(10):11169–11178, 2020.

[64] Alex Davies. Cisco pushes iot analytics to the extreme edge with mist computing. *Blog, Rethink Research. Available online: http://rethinkresearch. biz/articles/cisco-pushes-iot-analytics-extreme-edge-mist-computing-2 (accessed on 19 December 2014)*, 2014.

[65] Gary Davis. 2020: Life with 50 billion connected devices. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–1, 2018.

[66] Arka Daw, Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling, 2021.

[67] Kaikai Deng, Dong Zhao, Qiaoyue Han, Shuyue Wang, Zihan Zhang, Anfu Zhou, and Huadong Ma. Geryon: Edge assisted real-time and robust object detection on drones via mmwave radar and camera fusion. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), sep 2022.

[68] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y. Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[69] Moritz Diehl, Rishi Amrit, and James B. Rawlings. A lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, 56(3):703–707, 2011.

[70] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. Ieee, 2010.

[71] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.

[72] Eustace M Dogo, Abdulazeez Femi Salami, Clinton O Aigbavboa, and Thembinkosi Nkonyana. Taking cloud computing to the extreme edge: A review of mist computing for smart cities and industry 4.0 in africa. *Edge computing: from hype to reality*, pages 107–132, 2019.

[73] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD00. ACM, August 2000.

[74] Carlos M. J. M. Dourado, Suane Pires P. da Silva, Raul Victor M. da Nobrega, Pedro P. Rebouças Filho, Khan Muhammad, and Victor Hugo C. de Albuquerque. An open ioht-based deep learning framework for online medical image recognition. *IEEE Journal on Selected Areas in Communications*, 39(2):541–548, 2021.

[75] D. Duchamp, S.K. Feiner, and G.Q. Maguire. Software technology for wireless mobile computing. *IEEE Network*, 5(6):12–18, 1991.

[76] Andrej Dyck, Ralf Penners, and Horst Lichter. Towards definitions for release engineering and devops. In *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, pages 3–3, 2015.

[77] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, March 2004.

[78] Khalid A. Eldrandaly, Mohamed Abdel-Basset, and Laila A. Shawky. Internet of spatial things: A new reference model with insight analysis. *IEEE Access*, 7:19653–19669, 2019.

[79] General Electric. What is edge computing? Blog post.

[80] Ziv Epstein, Aaron Hertzmann, Investigators of Human Creativity, Memo Akten, Hany Farid, Jessica Fjeld, Morgan R Frank, Matthew Groh, Laura Herman, Neil Leach, et al. Art and the science of generative ai. *Science*, 380(6650):1110–1111, 2023.

[81] EU. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). 2016.

[82] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[83] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. A knowledge compilation map for ordered real-valued decision diagrams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[84] Moufida Feknous, Thierry Houdoin, Bertrand Le Guyader, Joseph De Biasio, Annie Gravey, and Jose Alfonso Torrijos Gijón. Internet traffic analysis: A case study from two major european operators. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2014.

[85] George H. Forman and John Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, 1994.

[86] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics amp; Data Analysis*, 38(4):367–378, February 2002.

[87] T Fukushima, T Konno, K Kiyoyama, M Murugesan, K Sato, W-C Jeong, Y Ohara, A Noriki, S Kanno, Y Kaiho, et al. New heterogeneous multi-chip module integration technology using self-assembly method. In *2008 IEEE International Electron Devices Meeting*, pages 1–4. IEEE, 2008.

[88] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971, 2020.

[89] Péter Galambos. Cloud, fog, and mist computing: Advanced robot applications. *IEEE Systems, Man, and Cybernetics Magazine*, 6(1):41–45, 2020.

[90] Leonardo Galteri, Marco Bertini, Lorenzo Seidenari, and Alberto Del Bimbo. Video compression for object detection algorithms. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3007–3012, 2018.

[91] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition, 1994.

[92] John Gantz, David Reinsel, et al. Extracting value from chaos. *IDC iview*, 1142(2011):1–12, 2011.

[93] Hang Gao, Ningxin He, and Tiegang Gao. SVeriFL: Successive verifiable federated learning with privacy-preserving. *Information Sciences*, 622:98–114, 2023.

[94] Ying Gao, Yijian Chen, Xiping Hu, Hongliang Lin, Yangliang Liu, and Laisen Nie. Blockchain based iiot data sharing framework for sdn-enabled pervasive edge computing. *IEEE Transactions on Industrial Informatics*, 17(7):5041–5049, July 2021.

[95] Vinay B. Gavirangaswamy, Gagan Gupta, Ajay Gupta, and Rajeev Agrawal. Assessment of arima-based prediction techniques for road-traffic volume. In *Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems*, MEDES '13. ACM, October 2013.

[96] Cristinel Gavrila, Vlad Popescu, Mauro Fadda, Matteo Anedda, and Maurizio Murroni. On the suitability of hbbtv for unified smart home experience. *IEEE Transactions on Broadcasting*, 67(1):253–262, 2021.

[97] Patrick P Gelsinger. Microprocessors for the new millennium: Challenges, opportunities, and new frontiers. In *2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC (Cat. No. 01CH37177)*, pages 22–25. IEEE, 2001.

[98] Saeid Ghafouri, Alireza Karami, Danial Bidekani Bakhtiarvan, Aliakbar Saleh Bigdeli, Sukhpal Singh Gill, and Joseph Doyle. Mobile-kube: Mobility-aware and energy-efficient service orchestration on kubernetes edge servers. In *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. IEEE, December 2022.

[99] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[100] Shreya Ghosh, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya. Mobi-iost: Mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. *IEEE Transactions on Network Science and Engineering*, 7(4):2271–2285, 2020.

[101] Fabio Giust, Gianluca Verin, Kiril Antevski, Joey Chou, Yonggang Fang, Walter Featherstone, Francisco Fontes, Danny Frydman, Alice Li, Antonio Manzalini, et al. Mec deployments in 4g and evolution towards 5g. *ETSI White paper*, 24(2018):1–24, 2018.

[102] Nelson Mimura Gonzalez, Walter Akio Goya, Rosangela de Fatima Pereira, Karen Langona, Erico Augusto Silva, Tereza Cristina Melo de Brito Carvalho, Charles Christian Miers, Jan-Erik Mångs, and Azimeh Sefidcon. Fog computing: Data analytics and cloud distributed processing on the network edges. In *2016 35th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–9. IEEE, 2016.

[103] Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex online problems. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st*

International Conference on Machine Learning, volume 32 of *Proceedings of Machine Learning Research*, pages 100–108, Bejing, China, 22–24 Jun 2014. PMLR.

[104] Kannan Govindan. How artificial intelligence drives sustainable frugal innovation: A multitheoretical perspective. *IEEE Transactions on Engineering Management*, 71:638–655, 2024.

[105] Dayan Guan, Jiaxing Huang, Aoran Xiao, Shijian Lu, and Yanpeng Cao. Uncertainty-aware unsupervised domain adaptation in object detection. *IEEE Transactions on Multimedia*, 24:2502–2514, 2022.

[106] Anneli Guthke. Defensible model complexity: A call for data-based and goal-oriented model choice. *Groundwater*, 55(5):646–650, 2017.

[107] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. *Machine Learning*, 46(1/3):389–422, 2002.

[108] Sebastian Haag and Reiner Anderl. Digital twin – proof of concept. *Manufacturing Letters*, 15:64–66, January 2018.

[109] Karim Habak, Ellen W Zegura, Mostafa Ammar, and Khaled A Harras. Workload management for dynamic mobile device clusters in edge femtoclouds. In *Proceedings of the second ACM/IEEE symposium on edge computing*, pages 1–14, 2017.

[110] Pengzhan Hao, Yongshu Bai, Xin Zhang, and Yifan Zhang. Edgecourier: an edge-hosted personal service for low-bandwidth document synchronization in mobile cloud storage services. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–14, 2017.

[111] Trevor J. Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost . *Statistics and Its Interface*, 2:349–360, 2009.

[112] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[113] Xiaofei He, Ming Ji, and Hujun Bao. A unified active and semi-supervised learning framework for image compression. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72, 2009.

[114] J. Matthew Helm, Andrew M. Swiergosz, Heather S. Haeberle, Jaret M. Karnuta, Jonathan L. Schaffer, Viktor E. Krebs, Andrew I. Spitzer, and Prem N. Ramkumar.

Machine learning and artificial intelligence: Definitions, applications, and future directions. *Current Reviews in Musculoskeletal Medicine*, 13(1):69–76, January 2020.

[115] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[116] Samer Henry, Ahmed Alsohaily, and Elvino S Sousa. 5g is real: Evaluating the compliance of the 3gpp 5g new radio system with the itu imt-2020 requirements. *IEEE Access*, 8:42828–42840, 2020.

[117] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[118] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[119] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[120] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[121] Wei-Jian Hu, Jie Fan, Yong-Xing Du, Bao-Shan Li, Naixue Xiong, and Ernst Bekkering. Mdfc–resnet: An agricultural iot system to accurately recognize crop diseases. *IEEE Access*, 8:115287–115298, 2020.

[122] Ye Hu, Johnny Wong, Gabriel Iszlai, and Marin Litoiu. Resource provisioning for cloud computing. In *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pages 101–111, 2009.

[123] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[124] Dong Huang, Ping Wang, and Dusit Niyato. A dynamic offloading algorithm for mobile computing. *IEEE Transactions on Wireless Communications*, 11(6):1991–1995, 2012.

[125] Dong Huang, Ping Wang, and Dusit Niyato. A dynamic offloading algorithm for mobile computing. *IEEE Transactions on Wireless Communications*, 11(6):1991–1995, 2012.

[126] J-P Hubaux, Thomas Gross, J-Y Le Boudec, and Martin Vetterli. Toward self-organized mobile ad hoc networks: The terminodes project. *IEEE Communications Magazine*, 39(1):118–124, 2001.

[127] Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond*, pages 1–5, 2010.

[128] Wang Huifeng, Seifedine Nimer Kadry, and Ebin Deni Raj. Continuous health monitoring of sportsperson using iot devices based wearable technology. *Computer Communications*, 160:588–595, July 2020.

[129] Shao-Chou Hung, Hsiang Hsu, Shao-Yu Lien, and Kwang-Cheng Chen. Architecture harmonization between cloud radio access networks and fog networks. *IEEE Access*, 3:3019–3034, 2015.

[130] Randall J Hunt, John Doherty, and Matthew J Tonkin. Are models too simple? arguments for increased parameterization. *Groundwater*, 45(3):254–262, 2007.

[131] Shadi Ibrahim, Hai Jin, Bin Cheng, Haijun Cao, Song Wu, and Li Qi. Cloudlet: towards mapreduce implementation on virtual machines. In *Proceedings of the 18th ACM international symposium on High performance distributed computing*, pages 65–66, 2009.

[132] J. B. Beutel *et al.* Flower: A Friendly Federated Learning Research Framework, 2020.

[133] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[134] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2220–2228. IEEE, 2014.

[135] C. Jennings, Z. Shelby, J. Arkko, A. Keranen, and C. Bormann. Sensor measurement lists (senml). RFC 8428, RFC Editor, August 2018.

[136] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, Imyhxy, , Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing,

UnglvKitDe, Victor Sonck, Tkianai, YxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - yolov5 sota realtime instance segmentation, 2022.

[137] Krishna P Kadiyala and Jorge A Cobb. Inter-as traffic engineering with sdn. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7. IEEE, 2017.

[138] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable Federated Learning for Mobile Networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[139] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[140] Kuljeet Kaur, Sahil Garg, Gagangeet Singh Aujla, Neeraj Kumar, Joel J. P. C. Rodrigues, and Mohsen Guizani. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Communications Magazine*, 56(2):44–51, February 2018.

[141] Ruimin Ke, Yifan Zhuang, Ziyuan Pu, and Yinhai Wang. A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices. *IEEE Transactions on Intelligent Transportation Systems*, 22(8):4962–4974, 2021.

[142] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal. *Cuckoo: A Computation Offloading Framework for Smartphones*, page 59–79. Springer Berlin Heidelberg, 2012.

[143] Jing Yang Koh, Ido Nevat, Derek Leong, and Wai-Choong Wong. Geo-spatial location spoofing detection for internet of things. *IEEE Internet of Things Journal*, 3(6):971–978, 2016.

[144] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

[145] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[146] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018.

[147] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, 2010.

[148] Emrecan Kutay and Aylin Yener. Semantic text compression for classification. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1368–1373, 2023.

[149] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient Federated Learning via Guided Participant Selection. In *15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021*, pages 19–35. USENIX Association, 2021.

[150] Yung-Te Lai, Massoud Pedram, and Sarma B. K. Vrudhula. Bdd based decomposition of logic functions with application to fpga synthesis. In *Proceedings of the 30th international on Design automation conference - DAC '93*, DAC '93. ACM Press, 1993.

[151] Björn Leander, Aida Čaušević, and Hans Hansson. Applicability of the iec 62443 standard in industry 4.0 / iiot. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19. ACM, August 2019.

[152] Khaled B. Letaief, Yuanming Shi, Jianmin Lu, and Jianhua Lu. Edge artificial intelligence for 6g: Vision, enabling technologies, and applications. *IEEE Journal on Selected Areas in Communications*, 40(1):5–36, January 2022.

[153] Khaled B. Letaief, Yuanming Shi, Jianmin Lu, and Jianhua Lu. Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications. *IEEE Journal on Selected Areas in Communications*, 40(1):5–36, 2022.

[154] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[155] Xiaomin Li, Di Li, Jiafu Wan, Chengliang Liu, and Muhammad Imran. Adaptive transmission optimization in sdn-based industrial internet of things with edge computing. *IEEE Internet of Things Journal*, 5(3):1351–1360, June 2018.

[156] Yuanjiang Li, Yunfeng Chen, Kai Zhu, Cong Bai, and Jinglin Zhang. An Effective Federated Learning Verification Strategy and Its Applications for Fault Diagnosis in Industrial IoT Systems. *IEEE Internet of Things Journal*, 9(18):16835–16849, 2022.

[157] Jacky Liang, Viktor Makoviychuk, Ankur Handa, Nuttapong Chentanez, Miles Macklin, and Dieter Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning, 2018.

[158] Shizhuang Lin, Jingyu Liu, and Yanjun Fang. Zigbee based wireless sensor networks and its applications in industrial. In *2007 IEEE international conference on automation and logistics*, pages 1979–1983. IEEE, 2007.

[159] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[160] Chuanhong Liu, Caili Guo, Yang Yang, and Nan Jiang. Adaptable semantic compression and resource allocation for task-oriented communications. *IEEE Transactions on Cognitive Communications and Networking*, pages 1–1, 2023.

[161] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Client-Edge-Cloud Hierarchical Federated Learning. In *IEEE ICC*, pages 1–6, 2020.

[162] Luning Liu, Xin Chen, Zhaoming Lu, Luhan Wang, and Xiangming Wen. Mobile-edge computing framework with data compression for wireless network in energy internet. *Tsinghua Science and Technology*, 24(3):271–280, 2019.

[163] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[164] Sin Kit Lo, Yue Liu, Qinghua Lu, Chen Wang, Xiwei Xu, Hye-Young Paik, and Liming Zhu. Towards Trustworthy AI: Blockchain-based Architecture Design for Accountability and Fairness of Federated Learning Systems. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[165] Yunlong Lu, Xiaohong Huang, Ke Zhang, Sabita Maharjan, and Yan Zhang. Communication-efficient federated learning for digital twin edge networks in industrial iot. *IEEE Transactions on Industrial Informatics*, 17(8):5709–5718, August 2021.

[166] M. Abadi *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[167] Ratul Mahajan, John Zahorjan, and Brian Zill. Understanding wifi-based connectivity from moving vehicles. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 321–326, 2007.

[168] Somayeh Malakuti and Sten Grüner. Architectural aspects of digital twins in iiot systems. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, ECSA '18. ACM, September 2018.

[169] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, 2016.

[170] Yoshitomo Matsubara, Sabur Baidya, Davide Callegaro, Marco Levorato, and Sameer Singh. Distilled split deep neural networks for edge-assisted real-time systems. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*. ACM, October 2019.

[171] Yoshitomo Matsubara, Davide Callegaro, Sameer Singh, Marco Levorato, and Francesco Restuccia. Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing. In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 337–346, 2022.

[172] Yoshitomo Matsubara and Marco Levorato. Neural compression and filtering for edge-assisted real-time object detection in challenged networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2272–2279. IEEE, 2021.

[173] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Comput. Surv.*, 55(5), dec 2022.

[174] Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. Sc2 benchmark: Supervised compression for split computing, 2023.

[175] Carlo Mazzocca, Nicolò Romandini, Matteo Mendula, Rebecca Montanari, and Paolo Bellavista. Truflaas: Trustworthy federated learning as a service. *IEEE Internet of Things Journal*, 10(24):21266–21281, 2023.

[176] Andrew McAfee, Erik Brynjolfsson, Thomas H Davenport, DJ Patil, and Dominic Barton. Big data: the management revolution. *Harvard business review*, 90(10):60–68, 2012.

[177] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[178] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[179] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[180] Matteo Mendula and Paolo Bellavista. Energy-aware edge federated learning for enhanced reliability and sustainability. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 349–354, 2022.

[181] Matteo Mendula, Armir Bujari, Luca Foschini, and Paolo Bellavista. A data-driven digital twin for urban activity monitoring. In *2022 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2022.

[182] Matteo Mendula, Siavash Khodadadeh, Salih Safa Bacanli, Sharare Zehtabian, Hassam Ullah Sheikh, Ladislau Bölöni, Damla Turgut, and Paolo Bellavista. Interaction and behaviour evaluation for smart homes: Data collection and analytics in the scaledhome project. MSWiM ’20, page 225–233, New York, NY, USA, 2020. Association for Computing Machinery.

[183] Matteo Mendula, Sharon L.G. Cotreras, Marco Levorato, and Paolo Bellavista. Furcifer: a context adaptive middleware for real-world object detection exploiting local, edge, and split computing in the cloud continuum. 2024.

[184] Amilcar Meneses-Viveros, Erika Hernández-Rubio, Sonia Mendoza, José Rodríguez, and Ana Belem Márquez Quintos. Energy saving strategies in the design of mobile device applications. *Sustainable Computing: Informatics and Systems*, 19:86–95, 2018.

[185] Yang Meng, Ruonan Rao, Xin Zhang, and Pei Hong. Crupa: A container resource utilization prediction algorithm for auto-scaling based on time series analysis. In *2016 International Conference on Progress in Informatics and Computing (PIC)*, pages 468–472, 2016.

[186] Shin-ichi Minato and Saburo Muroga. *Binary Decision Diagrams*. CRC Press, December 1999.

[187] Behzad Mirkhanzadeh, Ali Shakeri, Chencheng Shao, Miguel Razo, Marco Tacca, Gabriele Maria Galimberti, Giovanni Martinelli, Marco Cardani, and Andrea Fumagalli. An sdn-enabled multi-layer protection and restoration mechanism. *Optical Switching and Networking*, 30:23–32, 2018.

[188] Bibhu Mishra, Ravi Hosur, Priya Nandihal, and Piyush Pareek. Cloud computing, emerging computing technology of new age. *International journal of health sciences*, 05 2022.

[189] Annette Möller and Jürgen Groß. Probabilistic temperature forecasting based on an ensemble autoregressive modification. *Quarterly Journal of the Royal Meteorological Society*, 142(696):1385–1394, March 2016.

[190] Roberto Morabito. Virtualization on internet of things edge devices with container technologies: A performance evaluation. *IEEE Access*, 5:8835–8850, 2017.

[191] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[192] National Institute of Standards and Technology (NIST). Mobile cloud computing.

[193] Bruce Jay Nelson. *Remote procedure call.* Carnegie Mellon University, 1981.

[194] Dinh C. Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N. Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H. Vincent Poor. Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges. *IEEE Internet of Things Journal*, 8(16):12806–12825, 2021.

[195] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.

[196] Ewa Nowara and Daniel McDuff. Combating the impact of video compression on non-contact vital sign measurement using supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[197] Jianli Pan and James McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, February 2018.

[198] Zhengyuan Pang, Lifeng Sun, Zhi Wang, Erfang Tian, and Shiqiang Yang. A survey of cloudlet based mobile computing. In *2015 International conference on cloud computing and big data (CCBD)*, pages 268–275. IEEE, 2015.

[199] Jaehyoung Park and Hyuk Lim. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences*, 12(2):734, January 2022.

[200] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[201] Mugen Peng, Shi Yan, Kecheng Zhang, and Chonggang Wang. Fog-computing-based radio access networks: Issues and challenges. *Ieee Network*, 30(4):46–53, 2016.

[202] Bigi Varghese Philip, Tansu Alpcan, Jiong Jin, and Marimuthu Palaniswami. Distributed real-time iot for autonomous vehicles. *IEEE Transactions on Industrial Informatics*, 15(2):1131–1140, 2019.

[203] Enrico Pomarico, Cédric Schmidt, Florian Chays, David Nguyen, Arielle Planchette, Audrey Tissot, Adrien Roux, Stéphane Pagès, Laura Batti, Christoph Clausen, Theo Lasser, Aleksandra Radenovic, Bruno Sanguinetti, and Jérôme Extermann. Statistical distortion of supervised learning predictions in optical microscopy induced by image compression. *Scientific Reports*, 12(1), March 2022.

[204] Francisco Prado, Marcel C. Minutolo, and Werner Kristjanpoller. Forecasting based on an ensemble autoregressive moving average - adaptive neuro - fuzzy inference system – neural network - genetic algorithm framework. *Energy*, 197:117159, April 2020.

[205] Jürgo S Preden, Kalle Tammemäe, Axel Jantsch, Mairo Leier, Andri Riid, and Emine Calis. The benefits of self-awareness and attention in fog and mist computing. *Computer*, 48(7):37–45, 2015.

[206] Qinglin Qi and Fei Tao. Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access*, 6:3585–3593, 2018.

[207] Haoran Qiu, Subho S. Banerjee, Saurabh Jha, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. FIRM: An intelligent fine-grained resource management framework for SLO-Oriented microservices. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 805–825. USENIX Association, November 2020.

[208] Majid Rabbani and Rajan Joshi. An overview of the jpeg 2000 still image compression standard. *Signal processing: Image communication*, 17(1):3–48, 2002.

[209] Waqas Ur Rahman, Choong Seon Hong, and Eui-Nam Huh. Edge computing assisted joint quality adaptation for mobile video streaming. *IEEE Access*, 7:129082–129094, 2019.

[210] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125 – 141, 2018.

[211] Anindita Raychaudhuri, Anwesha Mukherjee, Debashis De, and Sukhpal Singh Gill. *Green Internet of Things Using Mobile Cloud Computing: Architecture, Applications, and Future Directions*, page 213–229. Springer International Publishing, 2022.

[212] Andrea Reale. A guide to edge iot analytics. *International Business Machines, https://www. ibm. com/blogs/internet-of-things/edge-iot-analytics (accessed: 2021)*, 2017.

[213] Muhammad Habib ur Rehman, Ahmed Mukhtar Dirir, Khaled Salah, Ernesto Damiani, and Davor Svetinovic. TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT. *IEEE Transactions on Industrial Informatics*, 17(12):8485–8494, 2021.

[214] Jie Ren, Ling Gao, Xiaoming Wang, Miao Ma, Guoyong Qiu, Hai Wang, Jie Zheng, and Zheng Wang. Adaptive computation offloading for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–30, December 2021.

[215] Ju Ren, Yaoxue Zhang, Kuan Zhang, and Xuemin Shen. Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Communications Magazine*, 53(3):98–105, 2015.

[216] Pei Ren, Xiuquan Qiao, Yakun Huang, Ling Liu, Schahram Dustdar, and Junliang Chen. Edge-assisted distributed dnn collaborative computing approach for mobile web augmented reality in 5g networks. *IEEE Network*, 34(2):254–261, 2020.

[217] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[218] Dominik Riemer. Feeding the digital twin: Basics, models and lessons learned from building an iot analytics toolbox (invited talk). *2018 IEEE International Conference on Big Data (Big Data)*, pages 4212–4212, 2018.

[219] Fabiana Rossi, Valeria Cardellini, and Francesco Lo Presti. Elastic deployment of software containers in geo-distributed computing environments. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2019.

[220] Fabiana Rossi, Matteo Nardelli, and Valeria Cardellini. Horizontal and vertical scaling of container-based applications using reinforcement learning. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 329–338, 2019.

[221] Sunitha Safavat, Naveen Naik Sapavath, and Danda B Rawat. Recent advances in mobile edge computing and content caching. *Digital Communications and Networks*, 6(2):189–194, 2020.

[222] Ahmed Salem and Tamer Nadeem. Lamen: leveraging resources on anonymous mobile edge nodes. In *Proceedings of the Eighth Wireless of the Students, by the Students, and for the Students Workshop*, pages 15–17, 2016.

[223] Hani Sami, Azzam Mourad, Hadi Otrok, and Jamal Bentahar. Fscaler: Automatic resource scaling of containers in fog clusters using reinforcement learning. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1824–1829, 2020.

[224] Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *IEEE Communications Surveys & Tutorials*, 16(1):369–392, 2013.

[225] Mohamed Sarwat. Spatial data systems support for the internet of things: challenges and opportunities. *SIGSPATIAL Special*, 12(2):42–47, October 2020.

[226] Arjuna Sathiaseelan, Adisorn Lertsinsrubtavee, Adarsh Jagan, Prakash Baskaran, and Jon Crowcroft. Cloudrone: Micro clouds in the sky. In *Proceedings of the 2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, pages 41–44, 2016.

[227] Mahadev Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 1–7, 1996.

[228] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[229] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9, 2008.

[230] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, nov 2020.

[231] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, nov 2020.

[232] Sandra Scott-Hayward, Sriram Natarajan, and Sakir Sezer. A survey of security in software defined networks. *IEEE Communications Surveys & Tutorials*, 18(1):623–654, 2015.

[233] Syed Yousaf Shah, Zengwen Yuan, Songwu Lu, and Petros Zerfos. Dependency analysis of cloud applications for performance monitoring using recurrent neural networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1534–1543, 2017.

[234] Siroos Shahriari, Milad Ghasri, S. A. Sisson, and Taha Rashidi. Ensemble of arima: combining parametric and bootstrapping technique for traffic flow prediction. *Transportmetrica A: Transport Science*, 16(3):1552–1573, January 2020.

[235] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[236] Pradip Kumar Sharma, Mu-Yen Chen, and Jong Hyuk Park. A software defined fog node based distributed blockchain cloud architecture for iot. *IEEE Access*, 6:115–124, 2018.

[237] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[238] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[239] Robert H. Shumway and David S. Stoffer. *ARIMA Models*, page 75–163. Springer International Publishing, 2017.

[240] Murtaza Ahmed Siddiqi, Heejung Yu, and Jingon Joung. 5g ultra-reliable low-latency communication implementation challenges and operational issues with iot devices. *Electronics*, 8(9):981, 2019.

[241] Pedro M Pinto Silva, Joao Rodrigues, Joaquim Silva, Rolando Martins, Luís Lopes, and Fernando Silva. Using edge-clouds to reduce load on traditional wifi infrastructures and improve quality of experience. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 61–67. IEEE, 2017.

[242] Kiran Deep Singh and Sandeep K Sood. Qos-aware optical fog-assisted cyber-physical system in the 5g ready heterogeneous network. *Wireless Personal Communications*, 116(4):3331–3350, 2021.

[243] Sukhpal Singh and Inderveer Chana. Cloud resource provisioning: survey, status and future research directions. *Knowledge and Information Systems*, 49:1005–1069, 2016.

[244] Jack Sleuters, Yonghui Li, Jacques Verriet, Marina Velikova, and Richard Doornbos. A digital twin method for automated behavior analysis of large-scale distributed iot systems. In *2019 14th Annual Conference System of Systems Engineering (SoSE)*. IEEE, May 2019.

[245] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Byzantine-Resilient Secure Federated Learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2021.

[246] Eugene Y. Song, Martin Burns, Abhinav Pandey, and Thomas Roth. Ieee 1451 smart sensor digital twin federation for iot/cps research. In *2019 IEEE Sensors Applications Symposium (SAS)*. IEEE, March 2019.

[247] Borja Sotomayor, Rubén S Montero, Ignacio M Llorente, and Ian Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet computing*, 13(5):14–22, 2009.

[248] Vinicius Souza, Robson Cruz, Walmir Silva, Sidney Lins, and Vicente Lucena. A digital twin architecture based on the industrial internet of things technologies. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, January 2019.

[249] Charles Steinmetz, Achim Rettberg, Fabiola Goncalves C. Ribeiro, Greyce Schroeder, and Carlos E. Pereira. Internet of things ontology for digital twin in cyber physical systems. In *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, November 2018.

[250] Isamu Takai, Tomohisa Harada, Michinori Andoh, Keita Yasutomi, Keiichiro Kagawa, and Shoji Kawahito. Optical vehicle-to-vehicle communication system using led transmitter and camera receiver. *IEEE Photonics Journal*, 6(5):1–14, 2014.

[251] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[252] Zhiqing Tang, Xiaojie Zhou, Fuming Zhang, Weijia Jia, and Wei Zhao. Migration modeling and learning algorithms for containers in fog computing. *IEEE Transactions on Services Computing*, 12(5):712–725, 2019.

[253] David Tellez, Diederik Höppener, Cornelis Verhoef, Dirk Grünhagen, Pieter Nierop, Michal Drozdzal, Jeroen van der Laak, and Francesco Ciompi. Extending unsupervised neural image compression with supervised multitask learning. In Tal Arbel, Ismail Ben Ayed, Marleen de Bruijne, Maxime Descoteaux, Herve Lombaert, and Christopher Pal, editors, *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, volume 121 of *Proceedings of Machine Learning Research*, pages 770–783. PMLR, 06–08 Jul 2020.

[254] David Tellez, Geert Litjens, Jeroen van der Laak, and Francesco Ciompi. Neural image compression for gigapixel histopathology image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):567–578, 2021.

[255] Shreshth Tuli, Shikhar Tuli, Gurleen Wander, Praneet Wander, Sukhpal Singh Gill, Schahram Dustdar, Rizos Sakellariou, and Omer Rana. Next generation technologies for smart healthcare: challenges, vision, model, trends and future directions. *Internet Technology Letters*, 3(2), January 2020.

[256] Irshad Ullah and Shoaib Ur Rehman. Analysis of black hole attack on manets using different manet routing protocols, 2010.

[257] Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional?, 2017.

[258] Natesha B V and Ram Mohana Reddy Guddeti. Fog-based intelligent machine malfunction monitoring system for industry 4.0. *IEEE Transactions on Industrial Informatics*, 17(12):7923–7932, December 2021.

[259] Mohammad Hadi Valipour, Bavar Amirzafari, Khashayar Niki Maleki, and Negin Daneshpour. A brief survey of software architecture concepts and service oriented architecture. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 34–38, 2009.

[260] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, 2019.

[261] Wil MP Van der Aalst, Vladimir Rubin, HMW Verbeek, Boudewijn F van Dongen, Ekkart Kindler, and Christian W Günther. Process mining: a two-step approach to

balance between underfitting and overfitting. *Software & Systems Modeling*, 9:87–111, 2010.

[262] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition, 2008.

[263] Laura von Rueden, Jochen Garcke, and Christian Bauckhage. How does knowledge injection help in informed machine learning? In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2023.

[264] Raphael Wagner, Benjamin Schleich, Benjamin Haefner, Andreas Kuhnle, Sandro Wartzack, and Gisela Lanza. Challenges and potentials of digital twins and industry 4.0 in product design and production for high performance products. *Procedia CIRP*, 84:88–93, 2019.

[265] Jiafu Wan, Shenglong Tang, Zhaogang Shu, Di Li, Shiyong Wang, Muhammad Imran, and Athanasios Vasilakos. Software-defined industrial internet of things in the context of industry 4.0. *IEEE Sensors Journal*, page 1–1, 2016.

[266] Chih-Yu Wang and Hung-Yu Wei. IEEE 802.11n MAC enhancement and performance evaluation. *Mobile Networks and Applications*, 14(6):760–771, January 2009.

[267] Lizhe Wang, Jie Tao, Marcel Kunze, Alvaro Canales Castellanos, David Kramer, and Wolfgang Karl. Scientific cloud computing: Early definition and experience. In *2008 10th IEEE International Conference on High Performance Computing and Communications*, pages 825–830, 2008.

[268] Jonathan Stuart Ward and Adam Barker. Undefined by data: A survey of big data definitions, 2013.

[269] Jonathan Stuart Ward and Adam Barker. Undefined by data: a survey of big data definitions. *arXiv preprint arXiv:1309.5821*, 2013.

[270] Jun Wei, Tao He, and Tao Huang. Challenges of communication in mobile computing. In *Proceedings Technology of Object-Oriented Languages. TOOLS 27 (Cat. No. 98EX224)*, pages 196–203. IEEE, 1998.

[271] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[272] Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos. Fog orchestration for internet of things services. *IEEE Internet Computing*, 21(2):16–24, 2017.

[273] Jeremy T White. Forecast first: An argument for groundwater modeling in reverse. *Groundwater*, 55(5):660–664, 2017.

[274] Julian Wörmann, Daniel Bogdoll, Christian Brunner, Etienne Bührle, Han Chen, Evaristus Fuh Chuo, Kostadin Cvejoski, Ludger van Elst, Philip Gottschall, Stefan Griesche, Christian Hellert, Christian Hesels, Sebastian Houben, Tim Joseph, Niklas Keil, Johann Kelsch, Mert Keser, Hendrik Königshof, Erwin Kraft, Leonie Kreuser, Kevin Krone, Tobias Latka, Denny Mattern, Stefan Matthes, Franz Motzkus, Mohsin Munir, Moritz Nekolla, Adrian Paschke, Stefan Pilar von Pilchau, Maximilian Alexander Pintz, Tianming Qiu, Faraz Qureishi, Syed Tahseen Raza Rizvi, Jörg Reichardt, Laura von Rueden, Alexander Sagel, Diogo Sasdelli, Tobias Scholl, Gerhard Schunk, Gesina Schwalbe, Hao Shen, Youssef Shoeb, Hendrik Stapelbroek, Vera Stehr, Gurucharan Srinivas, Anh Tuan Tran, Abhishek Vivekanandan, Ya Wang, Florian Wasserrab, Tino Werner, Christian Wirth, and Stefan Zwicklbauer. Knowledge augmented machine learning with applications in autonomous driving: A survey, 2023.

[275] Feng Xia, Fangwei Ding, Jie Li, Xiangjie Kong, Laurence T. Yang, and Jianhua Ma. Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. *Information Systems Frontiers*, 16(1):95–111, October 2013.

[276] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51, 2014.

[277] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51, 2015.

[278] Xing Xie, Hua-Jun Zeng, and Wei-Ying Ma. Enabling personalization services on the edge. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 263–266, 2002.

[279] Minxian Xu, Chenghao Song, Shashikant Ilager, Sukhpal Singh Gill, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. Coscal: Multifaceted scaling of microservices with reinforcement learning. *IEEE Transactions on Network and Service Management*, 19(4):3995–4009, December 2022.

[280] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. esdnn: Deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology*, 22(3):1–24, August 2022.

[281] Yu Xu, Jianguo Yao, Hans-Arno Jacobsen, and Haibing Guan. Cost-efficient negotiation over multiple resources with reinforcement learning. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*. IEEE, June 2017.

[282] Yu Xu, Jianguo Yao, Hans-Arno Jacobsen, and Haibing Guan. Cost-efficient negotiation over multiple resources with reinforcement learning. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–6, 2017.

[283] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. St3d: Self-training for unsupervised domain adaptation on 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10368–10378, June 2021.

[284] Ming Yang and N. Bourbakis. An overview of lossless digital image compression techniques. In *48th Midwest Symposium on Circuits and Systems, 2005.*, pages 1099–1102 Vol. 2, 2005.

[285] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019.

[286] Zhaohui Yang, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. Energy Efficient Federated Learning Over Wireless Communication Networks. *IEEE Transactions on Wireless Communications*, 20(3):1935–1949, 2021.

[287] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5668–5675, July 2019.

[288] Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, and Sghaier Guizani. Mobile ad hoc cloud: A survey. *Wireless Communications and Mobile Computing*, 16(16):2572–2589, 2016.

[289] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78, 2015.

[290] Zhongzheng Yuan, Tommy Azzino, Yu Hao, Yixuan Lyu, Haoyang Pei, Alain Boldini, Marco Mezzavilla, Mahya Beheshti, Maurizio Porfiri, Todd E Hudson, et al. Network-aware 5g edge computing for object detection: augmenting wearables to “see” more, farther and faster. *IEEE Access*, 10:29612–29632, 2022.

[291] Haitao Zhang, Huadong Ma, Guangping Fu, Xianda Yang, Zhe Jiang, and Yangyang Gao. Container based video surveillance cloud service with fine-grained resource provisioning. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 758–765, 2016.

[292] Linquan Zhang, Zongpeng Li, and Chuan Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 433–441. IEEE, 2014.

[293] Xingzhou Zhang, Yifan Wang, Sidi Lu, Liangkai Liu, Lanyu xu, and Weisong Shi. Openei: An open framework for edge intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1840–1851, 2019.

[294] Bo Zhao, Zhi Xu, Caixia Chi, Sencun Zhu, and Guohong Cao. *Mirroring Smartphones for Good: A Feasibility Study*, page 26–38. Springer Berlin Heidelberg, 2012.

[295] Wei Zhao, Xuan Wang, Bozhao Qi, and Troy Runge. Ground-level mapping and navigating for agriculture based on iot and computer vision. *IEEE Access*, 8:221975–221985, 2020.

[296] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[297] Fenghua Zhu, Yisheng Lv, Yuanyuan Chen, Xiao Wang, Gang Xiong, and Fei-Yue Wang. Parallel transportation systems: Toward iot-enabled smart urban traffic control and management. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4063–4071, 2020.

Acknowledgements

I am immensely grateful for the invaluable guidance provided by my advisor, Paolo Bellavista. His exemplary teaching style, leading by example, gave me the right directions not only in the attempt to be a good researcher but also to become a better human being. Thank you for the patience and trust that I have been able to appreciate on numerous occasions.

Then, I cannot thank my family enough for their punctual, discreet, and respectful way of providing me with both emotional and practical support. Your belief in me, even during times of self-doubt, has been a constant source of resilience.

I would also like to express my gratitude to the members of Bologna's research groups. The stimulating environment within the Middleware and Artificial Intelligence communities provided me with inspiration and collaborative support far beyond what I could have imagined.

But the most important person I am happy to thank is certainly and always my Bubu.
For your big heart and your strength. For the patience and the spontaneity.
And for re-teaching me the beauty of small things.
Thank you.