Emerging Trends in Machine Intelligence and Big Data

AI-Driven Task Scheduling in Heterogeneous Fog Computing Environments: Optimizing Task Placement Across Diverse Fog Nodes by Considering Multiple QoS Metrics

Kaushik Sathupadi[1]

[1]*Staff Engineer, Google LLC, Sunnyvale, CA*
ORCID: 0009-0007-1189-2293

This manuscript was compiled on Dec 3, 2020

Abstract

Fog computing paradigm extends cloud services to the network's edge in order to improve the performance of latency-sensitive and resource-constrained applications by enabling localized processing and storage. As the proliferation of Internet of Things (IoT) devices continues to grow, fog computing presents a viable solution for addressing the limitations of centralized cloud architectures. However, effective task scheduling in fog computing environments is a significant challenge in heterogeneous networks where nodes vary in terms of computational power, storage, and energy capacity. This paper explores the development and application of AI-based algorithms for task scheduling in heterogeneous fog computing environments, focusing on optimizing QoS metrics simultaneously. The paper first provides an overview of fog computing architecture and the challenges posed by heterogeneity. It then explores the role of AI in task scheduling, highlighting how machine learning (ML) and reinforcement learning (RL) techniques can be leveraged to improve decision-making processes. Through a detailed analysis of existing AI-based scheduling models, this study outlines the key features, advantages, and limitations of each. The paper also proposes a novel AI-driven approach that integrates multiple QoS metrics, evaluates task placement efficiency, and adapts to real-time network conditions. The findings suggest that AI-driven scheduling can significantly reduce latency, minimize energy consumption, and improve response times, contributing to more efficient resource utilization in fog computing environments.

Keywords: *AI-based task scheduling, Fog computing, Heterogeneous networks, Internet of Things (IoT), Latency reduction, Machine learning (ML), Quality of Service (QoS)*

Received: June 10, 2020 **Revised:** September, 16, 2020 **Accepted:** December, 2, 2020 **Published:** December, 10, 2020

1. Introduction

Smartphones, sensors, and IoT devices continuously generate large amounts of data, but processing this data to extract insights demands significant computational resources. Due to the limited processing power, memory, and battery life of these devices, they are unable to handle such tasks efficiently. Consequently, the processing is offloaded to more powerful systems, as attempting to perform it locally would be impractical or ineffective.

One common solution to this resource limitation is the use of cloud services for computational offloading. In this scenario, data are transmitted from the end device to an application running in a cloud data center, where the processing occurs. Cloud computing offers several advantages, notably its virtually unlimited computational resources, which can handle complex tasks and large-scale data processing without being restricted by local device limitations. This makes cloud-based offloading a viable solution in many contexts where the data volume or processing demand is high. However, this approach comes with its own set of challenges in scenarios where latency and bandwidth are critical considerations.

Latency is a key concern when offloading tasks to a cloud data center, as the center may be located far from the end device, possibly across multiple network hops. In application domains that demand real-time or near-real-time data processing—such as augmented reality, road traffic control systems, and gaming—such latency can be a critical barrier to usability. In these cases, even small delays caused by network transmission times and data center processing overhead can make cloud-based solutions impractical.

Additionally, large-scale data uploads to the cloud introduce the potential for network congestion, especially if many devices are simultaneously transmitting vast quantities of data to the same cloud infrastructure. As the number of connected devices continues to grow exponentially, this problem becomes more pronounced, leading to bottlenecks in network traffic that not only degrade the performance of individual applications but also increase overall system latency. This can further exacerbate delays and reduce the effectiveness of real-time applications. In situations where both high bandwidth and low latency are essential, cloud-based offloading may fail to meet the stringent requirements for system responsiveness and performance.

To address the challenges inherent in cloud computing those related to latency and network congestion, the paradigm of fog computing has emerged as a complementary approach. Fog computing introduces an intermediary computational layer between the end devices and the cloud [1]. This intermediary layer is composed of fog nodes, which are computational resources deployed in a geographically distributed manner, typically near the edge of the network, closer to the end devices. By positioning fog nodes closer to the data sources, fog computing can significantly reduce the latency associated with data transmission, as the data no longer need to travel to distant cloud data centers for processing. Fog computing represents a layered architecture designed to enable ubiquitous access to a shared, scalable continuum of computational resources. This paradigm is well-suited for the deployment of distributed, latency-sensitive applications and services within the Internet of Things (IoT) ecosystem. The core idea behind fog computing is to bring computational, storage, and network resources closer to the edge, reducing the reliance on centralized cloud services by positioning intermediate fog nodes between end-devices and the cloud. These fog nodes, which can be either physical or virtual, manage and process data locally or, when

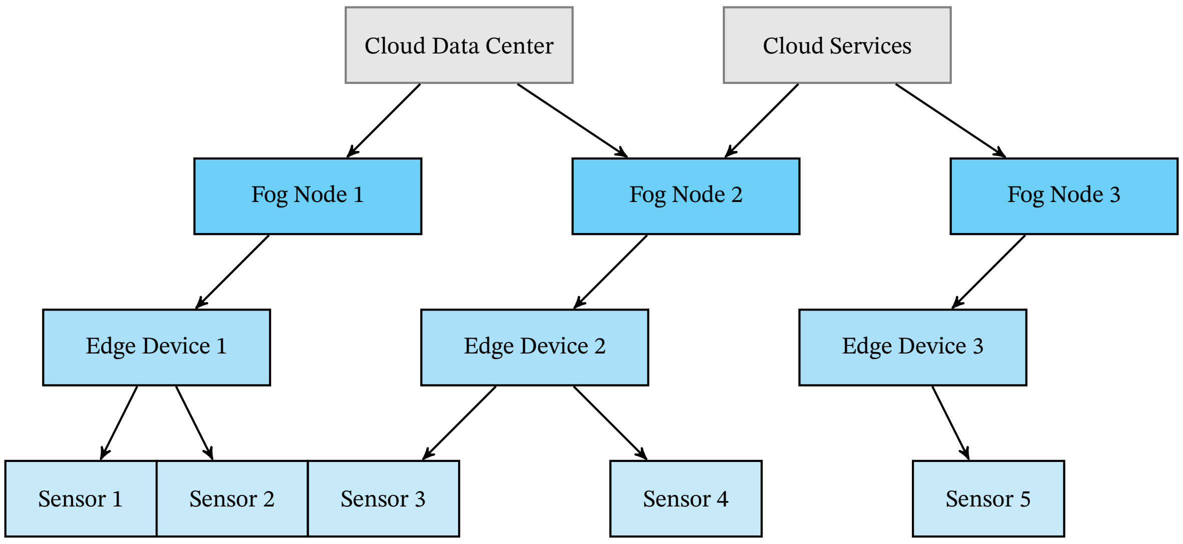


Figure 1. Fog Computing Architecture

Characteristic	**Definition**	**Importance**	**Example Use Case**
Contextual Location Awareness	Awareness of fog nodes' logical positioning	Minimizes latency by optimizing communication paths	Applications requiring low-latency, such as real-time health monitoring
Geographical Distribution	Distribution of computing resources across multiple locations	Supports applications that require distributed resources	High-quality video streaming for vehicles in motion
Heterogeneity	Ability to process data from various sources with different network capabilities	Facilitates data handling across diverse devices and networks	Gathering and processing data from smart cities or industrial IoT systems
Interoperability and Federation	Collaboration among multiple service providers and systems	Ensures seamless integration of services across domains	Real-time streaming involving different networks and providers
Real-Time Interactions	Real-time processing as opposed to batch processing	Enables instant analysis and response to time-sensitive data	Autonomous vehicles or real-time traffic management
Autonomy	Ability of fog nodes to function independently and make decisions locally	Allows for localized decision-making, reducing dependence on central systems	Individual smart home systems or local manufacturing units

Table 1. Essential characteristics of fog computing and their significance.

necessary, forward it to the cloud for additional processing [2].

Fog nodes function similarly to cloud resources but are localized to the network edge, making them ideal for latency-sensitive applications. This proximity allows end devices to offload their computational tasks to a nearby fog node, which can process the data and return the results with much lower latency than would be possible with cloud computing. Because fog nodes are distributed and decentralized, they alleviate the congestion problems associated with sending all data to a central cloud infrastructure. The load is distributed across multiple fog nodes, reducing the likelihood of network bottlenecks and improving overall system scalability.

Another advantage of fog computing is its flexibility in leveraging existing network infrastructure. For instance, network devices like routers, gateways, or base stations, which are already deployed across the network to manage data transmission, can be enhanced with additional computational resources to act as fog nodes. This reuse of existing equipment reduces the need for deploying entirely new infrastructure, thereby lowering the cost and complexity of implementing fog computing solutions. Alternatively, dedicated fog nodes can be deployed specifically for the purpose of handling computational tasks at the network edge, providing a more robust and tailored solution for environments with high computational or latency demands [3].

Fog nodes, much like cloud servers, support virtualization and containerization technologies. These technologies enable multiple applications from different tenants to run simultaneously on the same physical hardware while maintaining isolation between them. This is crucial for multi-tenant environments where different users or applications may have varying security and resource requirements. Virtualization facilitates resource management, making it easier to allocate computing power dynamically based on demand, and enables rapid deployment, scaling, and shutdown of applications as needed. Containers, in particular, have become a popular solution for managing lightweight, efficient, and portable application environments that can be easily moved between fog and cloud nodes.

The architectural model of fog computing is well-suited for a variety of real-time applications. In addition to reducing latency and preventing cloud overloading, fog computing offers improved fault tolerance. Since fog nodes are distributed, failures in individual nodes or network segments have a limited impact on the overall system. The decentralized nature of fog computing makes it inherently more resilient to localized disruptions, as other nearby nodes can continue processing data and supporting the system in the event of a failure.

In practice, fog computing operates as an extension of the cloud, offering a multi-tiered computing model that spans from the end device (the edge) to fog nodes and, ultimately, to cloud data centers. By integrating fog computing into an existing cloud infrastructure, systems can take advantage of both low-latency local processing and the expansive computational power of the cloud for more resource-intensive tasks that are less sensitive to delays [4]. This hybrid approach provides a scalable, flexible solution for a wide range of applications.

Applications that benefit most from fog computing include those that require immediate processing and action, such as autonomous vehicles, industrial automation, and smart cities. In autonomous vehicles, for instance, the data generated by onboard sensors must be processed instantaneously to make driving decisions, such as detecting obstacles and adjusting speed. Sending this data to a distant cloud data center for processing would introduce unacceptable delays, whereas processing it on nearby fog nodes ensures that the vehicle can react in real-time. Similarly, in industrial automation, machinery equipped with sensors can offload real-time monitoring and control tasks to fog nodes, enabling faster decision-making and reduced downtime.

Smart cities, which integrate IoT devices to monitor traffic, energy consumption, and environmental conditions, also stand to gain from fog computing. In such environments, data from sensors dispersed throughout a city can be processed locally at fog nodes to ensure quick responses to dynamic conditions, such as adjusting traffic signals based on real-time traffic flow or responding to fluctuations in energy demand. This capability not only reduces latency but also conserves network bandwidth by processing data closer to its source.

The Role of Fog Nodes

At the heart of the fog computing architecture lies the fog node, which can take the form of either physical components such as routers, switches, and servers, or virtual entities such as virtualized switches, cloudlets, and virtual machines. These fog nodes are context-aware, meaning they possess an understanding of their geographic and logical position within the network, which is crucial for their ability to deliver low-latency services. They also provide essential communication and data management services, bridging the gap between the network's edge, where IoT devices operate, and the centralized cloud. In essence, fog nodes create a localized computing environment that enhances the overall efficiency and responsiveness of the network.

Fog nodes can be deployed in various configurations depending on the system's requirements. For instance, they can operate in a centralized manner, where one or a few nodes handle most of the computational tasks, or they can be part of a decentralized architecture, where multiple nodes collaborate to deliver services. In certain cases, fog nodes may operate as stand-alone units, communicating with one another to achieve distributed processing. Alternatively, they can form federated clusters that provide horizontal scalability by mirroring services or extending their reach across geographically dispersed locations. This ability to federate is key to the scalability and adaptability of fog computing, especially in large-scale IoT applications where devices are often spread over vast areas [5].

2. Characteristics of Fog Computing and Fog Nodes

Fog computing distinguishes itself from other computing paradigms like cloud or edge computing through several essential characteristics. However, not all applications or services utilize the full range of fog computing capabilities. One of its defining features is contextual location awareness and low latency, which allows fog nodes to process data locally or determine the shortest path for communication, thereby minimizing delays. This is crucial for latency-sensitive applications such as autonomous vehicles, smart manufacturing, and healthcare monitoring, where real-time data analysis and response are essential. The close proximity of fog nodes to smart end-devices ensures rapid processing, eliminating the need to rely on distant cloud services [6].

In contrast to the centralized nature of cloud computing, fog computing resources are deployed across wide geographical areas to support dynamic, mobile applications. For instance, fog nodes can be positioned along highways to deliver high-quality streaming services to vehicles in motion, maintaining seamless connectivity and service continuity. This distributed architecture ensures that fog computing can meet the demands of geographically diverse applications [7].

Heterogeneity is also a core feature of fog computing, as it supports a wide variety of devices and communication protocols. Fog nodes must be able to interact with different types of end-devices, ranging from basic sensors to complex machinery, while managing data across multiple communication technologies. This flexibility is important in IoT environments, where devices vary widely in terms of form factor, communication capabilities, and data formats.

Interoperability and federation are crucial for many fog computing applications, particularly those involving cooperation between different service providers. For example, real-time services like video streaming or remote monitoring often require seamless integration across multiple domains. To achieve this, fog computing must support interoperability between heterogeneous systems and facilitate the creation of federated architectures, where fog nodes from different providers collaborate and share resources.

Real-time interactions further set fog computing apart. Unlike batch processing, which is common in traditional cloud environments, fog computing supports applications that require immediate feedback. This capability is important in scenarios like industrial automation, emergency response systems, or telemedicine, where timely data processing and minimal delay are critical for effective operation.

To support these characteristics, fog nodes need to exhibit several key attributes. Autonomy is one such attribute, allowing fog nodes to operate independently and make local decisions without relying on a centralized controller. This autonomy enables low-latency operations and enhances scalability and resilience, as fog nodes within a cluster can coordinate to optimize resource usage and service delivery.

Heterogeneity in fog nodes is essential, as they must be capable of functioning in diverse environments and supporting a broad range of devices. In IoT ecosystems, fog nodes interact with a variety of devices, such as sensors, actuators, and cameras, each with unique communication protocols and data formats. Despite differences in computational power, storage capacity, and networking capabilities, fog nodes must operate cohesively within the fog computing framework [8].

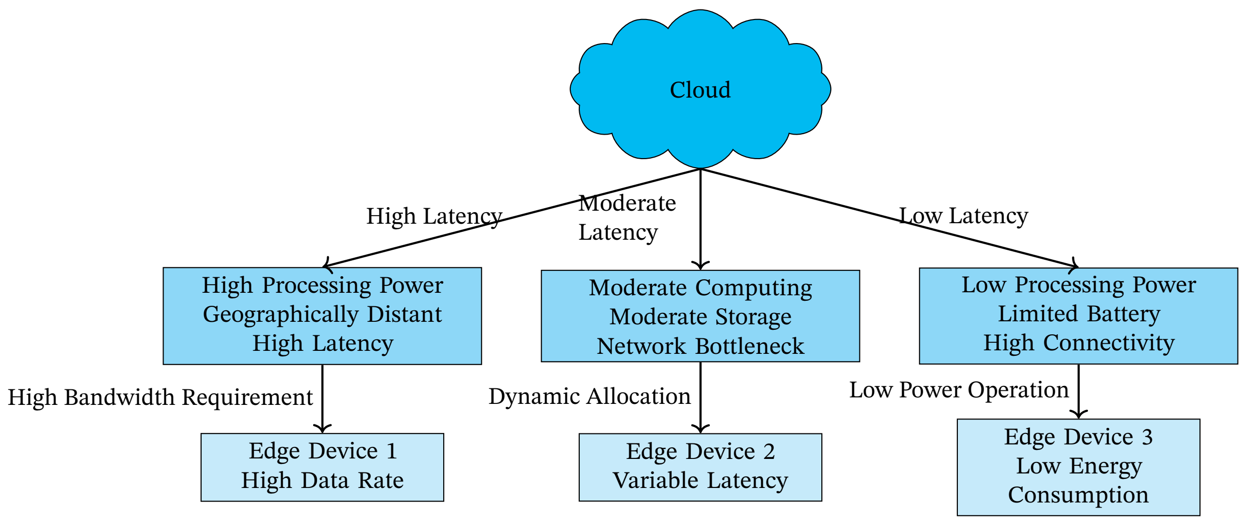
Hierarchical clustering is another important attribute of fog nodes. These nodes are often organized into hierarchical structures, with different layers providing various levels of service and processing capabilities. Lower layers might focus on real-time data processing close to the end-devices, while higher layers perform more complex analytics or serve as backups for lower-level nodes. This hierarchical arrangement ensures efficient resource allocation and enhances fault tolerance, as higher-level nodes can take over tasks if lower-level nodes fail.

Manageability is a critical attribute given the complexity of fog computing networks. Fog nodes are typically equipped with advanced management systems that automate routine tasks such as software updates, resource allocation, and fault detection. This reduces the operational overhead of managing large, distributed networks and ensures that resources are used efficiently. Programmability is a key feature of fog nodes, allowing them to be customized by various stakeholders, such as network operators, service providers, and end-users. This enables fog nodes to adapt to changing network conditions or application requirements dynamically. Programmability also facilitates the deployment of new services and applications without necessitating changes to the underlying hardware infrastructure, ensuring that fog computing remains flexible and scalable.

3. Background: Heterogeneous Fog Computing and QoS Metrics

3.1. Heterogeneous Fog Computing Environments

Fog computing has emerged as a vital paradigm in addressing the limitations of traditional cloud computing in the context of Internet of Things (IoT) applications. This decentralized model shifts data processing closer to the edge of the network, thereby reducing latency and bandwidth consumption, both of which are critical in scenarios where real-time processing is required. Unlike cloud computing, where data is typically transmitted to remote data centers for centralized processing, fog computing operates on geographically distributed nodes situated closer to the data source. These fog nodes play



This figure illustrates the heterogeneous nature of fog nodes in a fog computing environment. Nodes vary in processing power, energy constraints, and connectivity, leading to distinct challenges in scheduling and resource allocation. The diversity of fog nodes results in differentiated performance characteristics, affecting task execution efficiency. Task scheduling requires adaptive models to address varying latency, power consumption, and bandwidth demands.

Figure 2. Illustration of a Heterogeneous Fog Computing Environment and Its Challenges

a pivotal role in enabling real-time processing and decision-making in IoT ecosystems by bringing computation, storage, and networking resources near end users [9].

A key characteristic of fog computing environments is their inherent heterogeneity. The nodes in a fog network can range from high-performance edge servers with substantial computing resources to resource-constrained IoT devices with limited processing power, memory, and storage capacity. This heterogeneity stems from the diverse range of devices deployed across the network, each designed to serve specific purposes within the fog ecosystem. The spectrum of fog devices includes industrial control systems, smart sensors, mobile devices, and even high-end servers, all of which differ not only in hardware capabilities but also in their energy efficiency and communication protocols. For instance, a fog node located in a data center will typically have much higher processing and storage capabilities compared to a fog node embedded in a smart camera or a drone, which operates with stringent resource constraints. This variability in fog nodes complicates the task scheduling process, as an effective scheduling algorithm must intelligently match tasks to nodes based on the computational requirements of the task and the available resources of the nodes.

In addition to heterogeneity, fog computing environments are dynamic, meaning that the composition of the network is constantly changing. Fog nodes are often deployed in mobile or energy-constrained environments, and as a result, nodes can frequently join or leave the network, or experience fluctuations in available resources due to varying workloads or energy consumption. The mobility of fog nodes, such as those embedded in vehicles or drones, introduces complexity as the nodes can move in and out of communication range, leading to intermittent connectivity and changes in available bandwidth. Moreover, the dynamic nature of energy availability in battery-powered devices means that energy consumption must be carefully managed to avoid node failures or disruptions in service.

These dynamic characteristics require task scheduling algorithms to be adaptive and resilient. An optimal task scheduling algorithm must account not only for the computational capabilities and resource availability of the nodes at a given moment but also for the possibility of future changes in the network topology or resource availability. This adaptive approach ensures that the system can maintain high levels of service quality, even as the composition and capacity of the fog network change. For example, a task that is allocated to a resource-rich node initially might need to be migrated to a different node if the original node experiences an increase in workload or a reduction in available energy. Such flexibility is essential to maintaining the robustness and efficiency of the fog computing environment.

To address these challenges, the scheduling process must also consider multiple factors, such as the processing power, memory, and storage capacities of each fog node, as well as network-related parameters like bandwidth and latency. Scheduling decisions must balance the load across nodes to prevent resource overutilization, which could lead to performance bottlenecks, and underutilization, which would waste valuable computational resources. Additionally, the energy consumption of fog nodes those that are battery-powered, must be carefully managed to prolong the operational lifetime of the nodes and the overall network. Thus, the complexity of task scheduling in heterogeneous fog environments is significantly higher than in more homogeneous computing environments, such as traditional cloud computing, where resources are more predictable and standardized.

The heterogeneity of fog nodes necessitates the use of specialized algorithms capable of handling the diverse constraints and capabilities of the different devices within the network. Traditional task scheduling approaches, which typically assume a relatively uniform and stable set of resources, are insufficient for fog computing environments, as they fail to account for the variability and unpredictability of the fog nodes. Instead, advanced scheduling algorithms must leverage real-time information about the state of the network and the available resources of each node to make dynamic, context-aware decisions. Such algorithms often incorporate machine learning or optimization techniques to predict changes in the network and adapt the scheduling strategy accordingly [10] [9].

A common approach to dealing with heterogeneity and dynamism in fog computing is to employ a hierarchical task scheduling model, where tasks are first allocated to a high-level coordinator, which then distributes them to the most appropriate fog nodes. This allows the system to take a global view of the network and its resources, ensuring that tasks are allocated in a way that maximizes overall system efficiency. In some cases, multi-objective optimization techniques are used to balance multiple criteria, such as minimizing latency, maximizing resource utilization, and conserving energy, all while maintaining the quality of service (QoS) requirements for the appli-

Table 2. Comparison of Cloud and Fog Computing in IoT Environments

Feature	**Cloud Computing**	**Fog Computing**
Processing Location	Centralized data centers	Distributed, close to end devices
Latency	Higher (due to distance to cloud)	Lower (due to proximity to data sources)
Resource Scalability	High (virtually unlimited)	Limited (depends on local nodes)
Energy Consumption	High, but efficient in large-scale computations	Lower, but must account for energy-constrained devices
Network Dependence	Strong dependence on high-speed internet	Less dependent, as processing is done locally
Mobility Support	Limited	High (suitable for mobile environments)
Real-time Processing	Limited	High (ideal for real-time applications)

cation.

3.2. Quality of Service (QoS) Metrics

In fog computing environments, ensuring high quality of service (QoS) is crucial for maintaining the performance and reliability of applications those that require real-time processing and low-latency communication. Unlike traditional cloud computing, where QoS metrics are largely determined by the centralized infrastructure, fog computing introduces a more complex set of QoS challenges due to its decentralized and heterogeneous nature. Optimizing task placement in fog networks requires a multi-objective approach, as various QoS metrics must be considered simultaneously to ensure that the system meets the diverse requirements of different applications. These metrics often include latency, energy consumption, response time, and resource utilization, among others.

One of the primary motivations for adopting fog computing is the need to reduce latency, especially for applications that require immediate feedback. Latency refers to the delay between data generation at the edge of the network and the completion of the corresponding task processing. In cloud computing, data must be transmitted to distant data centers, leading to significant delays, especially in applications with strict real-time requirements, such as autonomous vehicle control, industrial automation, and augmented reality. Fog computing addresses this issue by processing data closer to the source, thereby minimizing the distance that data needs to travel and significantly reducing latency. Consequently, an effective task scheduling algorithm must prioritize minimizing latency for tasks that are latency-sensitive and require near-instantaneous responses.

Energy consumption is another critical QoS metric in fog computing environments, especially for battery-powered fog nodes that have limited energy resources. In many cases, fog nodes are deployed in locations where frequent recharging or replacement of batteries is impractical, such as remote sensors, drones, or mobile devices. As a result, minimizing energy consumption is essential to prolonging the operational lifetime of these nodes and ensuring the overall sustainability of the fog network. Task scheduling algorithms must carefully balance the trade-off between performance and energy efficiency, allocating tasks to nodes in a way that minimizes energy consumption while still meeting the performance requirements of the application. For example, energy-intensive tasks should be allocated to nodes with ample energy resources or those that are connected to a reliable power source, while less energy-demanding tasks can be assigned to battery-powered nodes to conserve energy.

Response time, which refers to the time taken for the system to respond to a user request, is closely related to both latency and processing time. In fog computing environments, response time can be impacted by several factors, including network latency, the computational delay experienced during task execution, and the availability of resources on the fog nodes. A well-designed task scheduling algorithm must ensure that tasks are executed in a timely manner to provide a satisfactory user experience for applications that require real-time interaction. This can be achieved by optimizing the placement of tasks on fog nodes with low latency and sufficient computational power, as well as by dynamically adjusting the allocation of tasks based on the current state of the network.

Resource utilization is another important QoS metric in fog computing environments. Efficient utilization of computational and storage resources is essential to preventing bottlenecks and ensuring a balanced load distribution across the network. In a heterogeneous fog environment, where nodes vary significantly in terms of their capabilities, it is important to ensure that resources are used efficiently to avoid both overutilization and underutilization. Overutilized nodes may experience performance degradation due to excessive workloads, while

underutilized nodes represent a wasted opportunity for improving system performance. Task scheduling algorithms must therefore strive to achieve a balanced distribution of tasks across the network, taking into account the capabilities and current workloads of each fog node.

3.3. Challenges in Task Scheduling

Task scheduling in fog computing refers to the process of assigning computational tasks to available fog nodes in a way that optimizes the use of resources while meeting the application's performance requirements [11]. Unlike in cloud computing, where resources are abundant and relatively homogeneous, the decentralized and heterogeneous nature of fog environments makes task scheduling significantly more complex. A fundamental challenge is the need to balance multiple objectives simultaneously, such as minimizing latency, optimizing energy consumption, and maximizing resource utilization. Given the variety of tasks with differing computational demands and the diversity of fog nodes with varying capabilities, an efficient task scheduling algorithm must consider several factors, including the available processing power, memory, storage, and current workload of each node. Additionally, the dynamic nature of fog networks, where nodes frequently join or leave and their resources fluctuate complicates the scheduling process, requiring real-time decision-making and adaptability.

At its core, task scheduling in fog environments involves three main components: task allocation, resource monitoring, and scheduling optimization. Task allocation refers to the process of determining which fog nodes should execute specific tasks, based on their current resource availability and the computational requirements of the tasks. This component is critical in ensuring that tasks are distributed in a way that maximizes the overall system performance while preventing individual nodes from becoming overloaded. Resource monitoring is another essential component, as it involves continuously tracking the status of fog nodes, including their energy levels, CPU utilization, and network connectivity. Accurate and real-time resource monitoring is crucial for making informed scheduling decisions, as it allows the system to adapt to changes in the network and avoid allocating tasks to nodes that are already under heavy load or experiencing connectivity issues. Lastly, scheduling optimization focuses on improving the overall efficiency of the task scheduling process by optimizing specific objectives, such as minimizing response time, maximizing resource utilization, or reducing energy consumption.

One of the primary challenges in fog computing task scheduling is that it is classified as an NP-hard problem, meaning that finding an optimal solution within a reasonable time frame is computationally infeasible for large-scale networks. Traditional heuristic-based

Table 3. Key QoS Metrics in Fog Computing Environments

QoS Metric	Description
Latency	Time delay between data generation and task completion, typically denoted as $\tau = t_{\text{completion}} - t_{\text{generation}}$
Energy Consumption	Amount of energy consumed by fog nodes during task execution, represented as $E = P \times t$, where P is the power consumption and t is the time
Response Time	Time taken for the system to respond to user requests, defined as $T_r = t_{\text{response}} - t_{\text{request}}$
Resource Utilization	Efficiency in utilizing computational and storage resources, modeled as $\rho = \frac{\text{Used Resources}}{\text{Total Available Resources}}$

Component	Defenition	Considerations	Impact on System Performance
Task Allocation	Determines which fog nodes should execute specific tasks based on resource availability and task requirements	Ensures balanced distribution of tasks across nodes to prevent overloading and optimize system performance	Maximizes system performance and prevents bottlenecks
Resource Monitoring	Continuously tracks the status of fog nodes, including energy levels, CPU utilization, and network connectivity	Requires accurate, real-time data to enable informed scheduling decisions and adaptation to changing network conditions	Improves scheduling decisions, reduces the risk of assigning tasks to overloaded nodes, and enhances system reliability
Scheduling Optimization	Focuses on optimizing the task scheduling process to meet specific objectives, such as minimizing response time or energy consumption	Considers various optimization goals, such as reducing latency, maximizing resource utilization, or minimizing energy use	Enhances system efficiency and ensures that key performance metrics are met

Table 4. Core components of task scheduling in fog environments.

scheduling algorithms, such as round-robin or first-come, first-served (FCFS), are commonly used in simpler computing environments. However, these methods fall short in the context of fog computing, as they do not consider the heterogeneity of nodes or the dynamic nature of resource availability. For instance, a round-robin approach, which assigns tasks in a cyclic manner, may allocate a computationally intensive task to a resource-constrained node, resulting in suboptimal performance and increased latency. Similarly, FCFS scheduling does not account for the varying computational requirements of tasks or the fluctuating resource capacities of fog nodes, leading to inefficient resource utilization and potential bottlenecks in the network. To address these limitations, AI-driven task scheduling solutions have gained prominence. Machine learning algorithms, in particular, offer a promising approach by enabling systems to learn from historical data, predict resource availability, and adapt to changing conditions in real time. Such algorithms can dynamically adjust scheduling decisions based on the current state of the network, leading to more efficient task allocation and better overall performance in heterogeneous fog environments.

4. AI-Driven Algorithms for Task Scheduling

4.1. AI-Based Heuristic Algorithms

Heuristic-based AI algorithms have long been foundational in the development of task scheduling approaches within the realm of fog computing. These algorithms operate on predefined rules or criteria that guide the decision-making process for task assignment and resource allocation. Their early adoption stems from their relative simplicity and the ease with which they can be implemented, as compared to more sophisticated machine learning (ML) or reinforcement learning (RL) methods. The key feature of heuristic algorithms is their reliance on well-defined heuristics—rules of thumb or problem-solving strategies—that enable them to provide satisfactory solutions to scheduling problems without necessarily guaranteeing optimality. Their suitability for fog computing lies in their ability to strike a balance between computational complexity and performance, making them ideal for environments where the overhead of more adaptive, data-driven methods may be prohibitive.

Among the most prominent heuristic algorithms used in fog computing are genetic algorithms (GA), simulated annealing (SA), and particle swarm optimization (PSO). These algorithms, while distinct in their operational mechanisms, share a common goal: to iteratively improve upon an initial set of solutions by exploring the solution space and refining task allocation over time. For instance, genetic algorithms draw inspiration from the process of natural evolution. They employ operations such as selection, crossover, and mutation to evolve a population of candidate solutions towards higher-quality task scheduling outcomes. The iterative nature of genetic algorithms makes them well-suited for environments where tasks and resources are relatively static or where the solution space is too vast for brute-force enumeration. However, the performance of genetic algorithms can degrade in highly dynamic environments, as the time required to evolve solutions may render the approach impractical for real-time decision-making [12].

Simulated annealing, another heuristic approach, is inspired by the physical process of annealing in metallurgy, where a material is heated and then gradually cooled to remove defects and optimize its structural properties. In the context of task scheduling, simulated annealing works by probabilistically accepting worse solutions early in the process to escape local optima, with the likelihood of accepting worse solutions decreasing as the algorithm progresses. This method is effective in avoiding premature convergence on suboptimal task schedules, but like genetic algorithms, it is best suited for relatively stable fog environments where task characteristics do not change rapidly. The computational cost associated with each iteration is manageable in small-scale systems, making simulated annealing a viable option for many fog computing scenarios. However, in environments characterized by frequent task arrivals or rapid fluctuations in resource availability, the cooling schedule—an essential parameter of simulated annealing—can be difficult to tune for optimal performance.

Particle swarm optimization is another widely used heuristic that simulates the social behavior of flocks of birds or schools of fish. In PSO, a population of particles explores the search space, with each particle adjusting its position based on its own experience and the experience of neighboring particles. The global best and local best solutions guide the swarm towards better task schedules. PSO is attractive due to its simplicity and its ability to converge quickly to high-quality solutions. It is especially effective when applied to multi-objective task scheduling problems where trade-offs between conflicting goals, such as minimizing task completion time and maximizing resource utilization, must be made. However, as with other heuristic approaches, PSO can struggle in highly dynamic environ-

Table 5. Performance Comparison of Heuristic-Based AI Algorithms in Fog Computing

Algorithm	Task Completion Time	Resource Utilization	Scalability
Genetic Algorithm	Medium	High	Low
Simulated Annealing	Medium	Medium	Low
Particle Swarm Optimization	Low	High	Medium

ments when the search space changes frequently due to fluctuating task loads or resource availability. PSO tends to converge quickly, it may prematurely settle on suboptimal solutions if not properly tuned in the presence of complex, multi-modal search spaces.

Unlike machine learning-based approaches, which can learn and adapt to new task patterns and resource conditions over time, heuristic methods are generally static, relying on fixed rules or strategies that may not be well-suited to changing environments. In fog computing scenarios, where the network topology and resource availability may shift frequently due to the distributed and heterogeneous nature of the system, the static nature of heuristic algorithms can limit their effectiveness. This limitation is pronounced in larger or more dynamic fog environments, where task scheduling decisions must be made in real time and must account for varying levels of resource contention, network latency, and energy consumption.

To better illustrate the performance characteristics of heuristic-based AI algorithms in fog computing, it is useful to consider their application in a small, controlled fog environment. Table 1 provides a comparison of task scheduling performance across several heuristic algorithms under typical fog computing conditions. The table shows that heuristic approaches offer a reasonable balance between scheduling accuracy and computational overhead, they do not scale well as the complexity of the environment increases.

In Table 5, we see that PSO generally performs best in terms of task completion time and resource utilization, though its scalability is only medium. Genetic algorithms, while also high in resource utilization, tend to suffer from poor scalability and moderate task completion times. Simulated annealing, on the other hand, offers a compromise between completion time and resource utilization but shares the scalability limitations of genetic algorithms. These results underscore the trade-offs inherent in heuristic-based approaches, where simplicity and ease of implementation come at the cost of reduced adaptability and scalability in more complex or dynamic environments.

The limited scalability of heuristic algorithms makes them more appropriate for smaller fog computing environments, such as those found in industrial settings, where the number of tasks and resources is relatively fixed, and where real-time scheduling demands are moderate. In such scenarios, the overhead of more adaptive AI-based algorithms may not be justified, and the simplicity of heuristic methods offers a practical alternative. However, as the size and complexity of the fog environment grow, the limitations of heuristic algorithms become more pronounced. For instance, in smart city applications, where fog nodes must handle a highly dynamic and unpredictable flow of tasks, the static nature of heuristic rules can result in suboptimal resource allocation and increased task completion times.

Moreover, heuristic algorithms often require extensive tuning of parameters to achieve good performance, especially in fog environments where tasks have diverse computational requirements and where network conditions can vary unpredictably. For example, in genetic algorithms, the choice of population size, crossover rate, and mutation rate can have a significant impact on the quality of the scheduling solution. Similarly, in simulated annealing, the cooling schedule and initial temperature must be carefully selected to balance exploration and exploitation. Poorly chosen parameters can lead to premature convergence or excessive computational overhead, both of which are undesirable in fog computing systems with limited computational resources. Parameter tuning, while effective in static or moderately dynamic environments, becomes increasingly difficult as the scale and dynamism of the fog environment increase.

To address these challenges, some hybrid approaches have been proposed that combine heuristic algorithms with more adaptive methods. For example, hybrid genetic algorithms have been developed that incorporate local search strategies to refine the solutions produced by the genetic algorithm. These hybrid methods aim to combine the global search capabilities of heuristic algorithms with the adaptability of local search or ML-based methods, thereby improving the quality of task scheduling decisions in dynamic fog environments. Another approach involves using heuristic algorithms to generate initial solutions, which are then refined using ML techniques that can learn from past task scheduling decisions. Such hybrid methods represent a promising avenue for improving the performance of heuristic algorithms in fog computing in environments characterized by high levels of variability and complexity [13].

Table 2 provides a summary of the trade-offs between heuristic, machine learning, and hybrid approaches for task scheduling in fog computing environments.

As shown in Table 6, heuristic algorithms offer low complexity but lack the adaptability required for dynamic environments. Machine learning-based algorithms, while highly adaptable, come with increased computational complexity, making them more suitable for larger fog environments with abundant computational resources. Hybrid approaches strike a balance, offering moderate adaptability and complexity, and are thus seen as a promising direction for future research in fog computing task scheduling.

4.2. Deep Learning for Task Scheduling

Task scheduling is inherently a challenging problem due to the need to optimize multiple, often conflicting, objectives, such as minimizing execution time, balancing computational load, and optimizing energy consumption, while simultaneously maintaining high quality of service (QoS). Deep learning models, with their ability to learn from vast amounts of data and capture nonlinear relationships, provide a robust framework for addressing these challenges.

Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown great potential in modeling complex scheduling problems. CNNs, traditionally used for image recognition tasks, can be adapted to task scheduling by leveraging their ability to identify spatial hierarchies and patterns within the data. For instance, tasks with different resource requirements and varying execution times can be represented as multi-dimensional matrices, with CNNs effectively capturing the dependencies and constraints inherent in scheduling. By learning these relationships, CNN-based models can generate efficient schedules that balance the computational load across available fog nodes while minimizing overall task completion time. The hierarchical nature of CNNs also allows these models to scale well in large, distributed systems, enabling the processing of vast amounts of data without sacrificing performance.

Recurrent neural networks their more advanced forms like long short-term memory (LSTM) networks and gated recurrent units (GRUs), are equally well-suited for scheduling problems that involve sequential dependencies. Task scheduling often requires consideration of past events or the order in which tasks arrive. RNNs are capable of maintaining a memory of previous inputs and can model time-series data, making them highly effective for scheduling tasks that are interdependent or that require adaptive decision-making over time. LSTM networks, for example, have been successfully employed in dynamic scheduling environments where the arrival of new tasks

Table 6. Comparison of Task Scheduling Approaches in Fog Computing

Approach	Adaptability	Complexity	Suitability for Dynamic Environments
Heuristic Algorithms	Low	Low	Poor
Machine Learning-Based Algorithms	High	High	Good
Hybrid Approaches	Medium	Medium	Moderate

Table 7. Comparison of Traditional and DL-based Scheduling Methods

QoS Metric	Traditional Scheduling	DL-based Scheduling
Latency	Moderate to High	Low
Resource Utilization	Suboptimal	High
Scalability	Limited	High
Adaptability to Changing Conditions	Low	High

or the availability of fog nodes changes over time. These models are able to retain information about prior scheduling decisions and adjust future actions accordingly, leading to improved system performance and resource utilization.

A key advantage of DL-based scheduling algorithms lies in their ability to optimize quality of service (QoS) metrics dynamically. Traditional scheduling methods, such as heuristic algorithms or rule-based systems, often require manual tuning and are limited in their ability to adapt to changing system conditions or workloads. In contrast, deep learning models can continuously learn and adjust their scheduling strategies based on feedback from the system. This adaptability is useful in environments characterized by large-scale data flows, such as Internet of Things (IoT) networks or distributed fog systems, where the volume of tasks and the availability of resources fluctuate unpredictably. By training on historical data and incorporating real-time feedback, DL-based algorithms can improve QoS by minimizing latency, optimizing bandwidth usage, and reducing energy consumption.

DL-based scheduling models are adept at handling the scale and complexity of modern distributed systems, where the volume of data can overwhelm traditional models. In environments with large-scale data flows, such as those found in fog computing, the ability to process and analyze vast amounts of data in real-time is crucial. CNNs and RNNs, with their deep architectures, are well-suited for such tasks, as they can learn to recognize patterns in large datasets that might be missed by conventional models. This is relevant when scheduling tasks across multiple fog nodes, where the optimal scheduling policy may depend on subtle relationships between task requirements, network latency, and node availability. Deep learning models, trained on large-scale datasets, can capture these relationships and make real-time scheduling decisions that optimize system performance.

Table 1 illustrates a comparison between traditional scheduling approaches and DL-based scheduling methods in terms of key QoS metrics, such as latency, resource utilization, and scalability. As seen in the table, DL-based methods consistently outperform traditional techniques, in environments characterized by high task arrival rates and large data volumes.

Another area where deep learning excels in task scheduling is in the optimization of energy consumption. In fog computing environments, energy efficiency is a critical concern, especially as the number of IoT devices and the scale of data processing continue to grow [14] [15]. DL-based models, those that incorporate reinforcement learning (RL) techniques, can dynamically adjust task scheduling to minimize energy consumption without compromising performance. RL-based approaches, when integrated with deep learning architectures, enable systems to learn optimal scheduling policies through trial and error, continuously refining their strategies to improve energy efficiency. By modeling the system as a Markov decision process, where each scheduling action impacts future states of the system, DL-based models can make energy-aware scheduling decisions that reduce overall power consumption.

In environments where the task requirements and system capabilities are highly heterogeneous, DL-based models also provide a significant advantage by learning to handle diverse workloads. Fog computing systems typically involve a wide range of tasks with varying computational demands, network latencies, and deadlines. Traditional scheduling algorithms often struggle to balance these competing demands, especially when the system needs to prioritize tasks based on their deadlines or resource requirements. Deep learning models, however, can be trained on diverse datasets that encompass a variety of task types and system configurations, allowing them to generate schedules that optimize resource utilization while meeting task-specific constraints. For instance, CNNs can be employed to classify tasks based on their resource requirements, while RNNs can sequence tasks in a way that maximizes throughput and minimizes bottlenecks.

The ability of DL-based models to generalize across different scheduling scenarios is another critical factor in their success. Unlike heuristic-based algorithms, which often require significant re-tuning when applied to new environments, DL models can be retrained or fine-tuned on new datasets to accommodate changes in system architecture or workload characteristics. This flexibility is particularly beneficial in fog and edge computing environments, where the infrastructure is often highly dynamic, with fog nodes being added or removed over time. By incorporating transfer learning techniques, DL-based scheduling models can be adapted to new environments with minimal retraining, thereby reducing the computational overhead associated with model deployment and maintenance.

Table 8 highlights some of the key characteristics of CNNs and RNNs in the context of task scheduling, providing a comparison of their strengths and limitations. As shown, while CNNs excel at capturing spatial dependencies and handling large-scale data, RNNs are effective at modeling temporal dependencies and sequential decision-making processes.

Table 8. Comparison of CNNs and RNNs for Task Scheduling

Feature	CNN	RNN
Handling Spatial Dependencies	High	Moderate
Handling Temporal Dependencies	Low	High
Scalability	High	Moderate
Sequential Decision-Making	Low	High
Training Complexity	Moderate	High

4.3. Reinforcement Learning for Multi-Objective Optimization

Reinforcement learning (RL) has emerged as a powerful tool for solving complex optimization problems, those that involve multiple conflicting objectives, as is the case in fog computing environments. In a fog network, tasks must be scheduled across heterogeneous nodes while optimizing several Quality of Service (QoS) metrics, such as latency, energy consumption, response time, and resource utilization.

The dynamic and distributed nature of fog computing, along with the diverse capabilities of fog nodes, makes traditional heuristic methods inadequate. RL offers a promising solution by enabling systems to learn optimal policies through interaction with the environment, thereby allowing for real-time, adaptive decision-making that continuously improves based on feedback from the system's performance.

At the core of reinforcement learning is the concept of an agent that interacts with an environment by taking actions to achieve a goal, typically represented as maximizing a cumulative reward function. In the context of fog computing, the RL agent is tasked with making scheduling decisions, such as assigning tasks to fog nodes based on the current state of the network. The state could include factors like node availability, resource utilization, and task requirements, while the actions correspond to allocating specific tasks to available nodes. The reward function is designed to reflect the multiple objectives of the system, such as minimizing latency and energy consumption, while ensuring that the response time remains within acceptable limits. By carefully designing the reward structure, RL algorithms can learn to prioritize different QoS metrics based on the system's real-time needs, making them ideal for multi-objective optimization in fog environments.

One of the key advantages of using reinforcement learning in fog computing is its ability to handle the trade-offs between conflicting objectives. For example, reducing latency often requires allocating tasks to nodes with low communication delay, but these nodes may be energy-constrained, which could reduce their operational lifetime. Conversely, tasks assigned to nodes with higher energy reserves may experience longer delays due to their distance from the data source. RL algorithms can effectively balance these trade-offs by adjusting task assignments dynamically as network conditions evolve. Unlike static scheduling algorithms, which make decisions based on predefined criteria, RL agents continuously learn from the environment and adapt their policies over time. This ability to adapt is especially valuable in fog computing environments, where network topology and resource availability can change frequently due to node mobility, fluctuating workloads, and varying energy levels.

Multi-agent reinforcement learning (MARL) approaches offer further enhancements to the scalability and efficiency of task scheduling in large-scale fog computing environments. In MARL, multiple RL agents operate simultaneously, each responsible for optimizing a subset of the overall system. For instance, one agent may focus on optimizing energy consumption by managing task assignments for energy-constrained nodes, while another agent may prioritize reducing latency by ensuring that time-sensitive tasks are executed on nodes with low communication delays. The agents work collaboratively, sharing information and learning from each other's actions to achieve a global optimization of the system's objectives. MARL approaches are well-suited for distributed fog environments, where the computational complexity of scheduling decisions can become prohibitive for a single centralized agent. By distributing the decision-making process across multiple agents, MARL enables more efficient scaling, ensuring that even large and complex fog networks can be optimized effectively without overwhelming the scheduling system [16].

In addition to scalability, MARL also enhances the robustness of task scheduling in fog environments. In highly dynamic networks, where nodes may frequently join or leave the system, single-agent RL approaches may struggle to keep up with the rapid changes in network topology and resource availability. By contrast, multi-agent systems can localize decision-making, allowing each agent to focus on a specific region of the network or a subset of tasks. This localization improves the system's ability to respond quickly to changes, as each agent operates independently while still contributing to the overall optimization process. MARL reduces the computational burden on any single agent, making it possible to handle the high degree of heterogeneity and dynamism that characterizes fog computing environments [17] [4].

4.4. Hybrid AI Approaches

Hybrid AI approaches represent a sophisticated and promising avenue for optimizing task scheduling in fog computing environments by combining the strengths of multiple AI techniques. Given the complexity and multi-dimensional nature of fog computing—where scheduling decisions must account for diverse QoS metrics such as latency, energy consumption, response time, and resource utilization—traditional AI methods often fall short in addressing all aspects of the problem effectively. Hybrid AI models capitalize on the complementary advantages of various algorithms, merging different learning and optimization techniques to deliver more robust, adaptive, and efficient solutions for real-time task scheduling in dynamic and heterogeneous fog environments [5].

One common form of hybrid AI in fog computing involves the integration of machine learning (ML) with reinforcement learning (RL). In such systems, machine learning models are used to predict key network metrics, such as task loads, resource availability, or network congestion, based on historical data. These predictions enable more informed decision-making by providing RL agents with valuable contextual information, allowing them to focus on making real-time, adaptive scheduling decisions. For instance, an ML model might forecast an impending spike in task load based on past trends or changes in user behavior, giving the RL agent advance notice to preemptively redistribute tasks or allocate additional resources. This collaboration between predictive ML and adaptive RL ensures that task scheduling can both anticipate future demands and respond dynamically to evolving network conditions, leading to more efficient and timely execution of tasks across the fog nodes.

In addition to ML-RL hybrids, other approaches combine deep learning (DL) with evolutionary algorithms such as genetic algorithms (GA) to further enhance the task scheduling process. Deep learning excels at identifying complex patterns in large datasets, making it highly effective for analyzing the vast and often unstructured data generated by fog networks, such as traffic patterns, node usage, or task execution histories. When used in conjunction with genetic algorithms, which are adept at optimizing solutions by mimicking the process of natural selection, hybrid models can achieve a powerful balance between exploration and exploitation in the scheduling process. Genetic algorithms are well-suited for large search spaces, where they can iteratively evolve scheduling strategies by selecting, mutating, and combining the best-performing solutions from previous iterations. By integrating deep learning's capacity for deep pattern recognition with the exploration-driven optimization of genetic algorithms, hybrid models can intelligently explore a wide range of potential scheduling strategies while continuously refining and improving their approach to meet the system's dynamic needs.

Such a hybrid deep learning-genetic algorithm (DL-GA) system might work as follows: the deep learning component analyzes historical task data and identifies patterns that correlate with optimal task placement, such as the most efficient resource utilization based on past performance metrics. This information is fed into the genetic algorithm, which explores various combinations of task assignments and resource allocations, guided by the insights provided by the deep learning model. Over time, the genetic algorithm evolves its scheduling strategy, learning which node-task pairings result in the best overall system performance in terms of latency, energy efficiency, and response time. By balancing exploration (i.e., trying new combinations of task allocations) and exploitation (i.e., refining already successful strategies), the hybrid DL-GA approach can effectively navigate the large scheduling of a fog computing environment.

Hybrid AI approaches also enable a more holistic treatment of the trade-offs involved in multi-objective optimization. In fog computing, optimizing one QoS metric often comes at the expense of another—such as prioritizing low-latency task execution, which may increase energy consumption in battery-powered fog nodes. By leveraging multiple AI techniques, hybrid models can better navigate these trade-offs. For example, reinforcement learning may prioritize immediate QoS needs, such as reducing response time for real-time applications, while a deep learning model might assess longer-term impacts, such as node energy depletion. In this scenario, a hybrid system would balance short-term scheduling decisions with long-term resource sustainability, achieving a more nuanced optimization than any single technique could accomplish on its own.

Additionally, hybrid AI approaches offer significant advantages in terms of scalability and adaptability in large and geographically dispersed fog networks. As the scale of the network grows, so too does the complexity of the task scheduling problem, with an increasing number of nodes, tasks, and variables to consider. Hybrid AI models those that integrate distributed learning techniques, can break down the problem into more manageable sub-problems. For example, machine learning models deployed at local fog clusters could predict local resource availability and task loads, while reinforcement learning agents operate globally to make high-level scheduling decisions that optimize the entire network. This division of labor between local and global optimization components enables hybrid models to scale effectively, ensuring that task scheduling remains efficient and responsive even as the network grows in size and complexity.

5. Proposed AI-Driven Task Scheduling Model

This section presents a novel AI-driven task scheduling model tailored to optimize multiple Quality of Service (QoS) metrics, specifically in heterogeneous fog computing environments. Fog computing is a decentralized computing architecture that extends cloud computing capabilities closer to the network edge. It is critical in real-time, latency-sensitive applications such as IoT (Internet of Things), smart cities, and autonomous systems. One of the most challenging aspects of fog computing is task scheduling, which must account for highly dynamic conditions, resource heterogeneity, and the necessity to balance multiple, often conflicting QoS objectives such as latency, energy consumption, and response time.

The proposed model leverages advanced artificial intelligence techniques, integrating machine learning for predictive scheduling, reinforcement learning for dynamic adaptation, and a hybrid AI-based multi-objective optimization approach. This structured approach ensures that all critical QoS parameters are simultaneously optimized without sacrificing system performance. The following sections discuss the technical details of the proposed model's key components: predictive task placement, dynamic adaptation, and multi-objective optimization.

Predictive Task Placement

The foundation of the proposed task scheduling model lies in its ability to predict future workloads and node availability accurately. In heterogeneous fog environments, resource availability fluctuates due to the dynamic nature of edge devices, network conditions, and varying task requirements. Therefore, the task placement decision must consider future conditions to prevent suboptimal allocations that can lead to increased latency or node overload.

To address this challenge, we employ machine learning algorithms, specifically time series forecasting models such as Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs). These models are effective in capturing temporal dependencies in data, allowing them to predict task loads and node availability over time. The training data for these models are gathered from historical system performance logs, including metrics like task execution time, CPU usage, network latency, and energy consumption.

The predictive task placement module processes the incoming task stream and the predicted resource availability at each node in the fog network. Based on this information, tasks are assigned to nodes that can handle the anticipated workload while minimizing QoS metrics such as latency and energy consumption. Specifically, the objective function used in the machine learning model aims to balance task placement by considering the following parameters: (i) the predicted latency for each task on the given node, (ii) the energy consumption incurred by executing the task, and (iii) the node's expected availability based on predicted workloads.

A key feature of this module is its use of a weighted heuristic function that prioritizes nodes with the lowest predicted latency and energy consumption. For example, if the network detects a task with high computational requirements, the system predicts which nodes will be underutilized in the near future and assigns the task accordingly. This minimizes the chance of node overload and reduces the likelihood of bottlenecks. By integrating predictive analytics into the task placement process, we can ensure that the scheduling system is proactive, rather than reactive, to changes in the network.

Dynamic Adaptation through Reinforcement Learning

While predictive task placement is vital, fog computing environments are highly dynamic, and the network conditions can change rapidly due to fluctuating workloads, network congestion, or node failures. To handle such variability, the proposed model integrates reinforcement learning (RL) agents to dynamically adapt task scheduling decisions in real-time. Reinforcement learning is well-suited for such scenarios, as it allows the system to learn optimal policies based on continuous feedback from the environment.

The RL agents in our model are designed to adjust task allocations by learning from past decisions and network states. They continuously monitor the system's performance metrics, such as response time and node utilization, and adjust task placement strategies accordingly. The reinforcement learning framework we adopt involves a Markov Decision Process (MDP), where the system's state represents the current resource availability and task queue, the action space consists of possible task allocation strategies, and the reward function is based on QoS improvements like reduced latency and minimized energy consumption.

At each time step, the RL agent evaluates the current system state and selects an action (task allocation) that maximizes the cumulative reward. If the network conditions change unexpectedly, such as a spike in task requests or a node failure, the RL agent quickly adapts the scheduling policy to redistribute tasks and avoid potential system degradation. Over time, the RL agent converges to an optimal scheduling policy that balances real-time responsiveness with long-term QoS optimization. By leveraging the adaptability of reinforcement learning, our model ensures that task scheduling remains efficient even in highly dynamic fog computing environments.

Multi-Objective Optimization

In fog computing, multiple QoS metrics must be optimized simultaneously, often with competing objectives such as minimizing latency, reducing energy consumption, and improving system response time. Traditional scheduling approaches often focus on optimizing a single QoS metric, which can lead to suboptimal results for the system as a whole. To address this, our proposed model employs a hybrid AI-based multi-objective optimization approach that balances all key QoS metrics.

The multi-objective optimization module operates on the outputs of the predictive task placement and dynamic adaptation layers, using a combination of techniques such as genetic algorithms and Pareto-based optimization to find the best trade-off between conflicting objectives. Genetic algorithms (GAs) are well-suited for this task due to their ability to explore a large search space and converge to an optimal or near-optimal solution over multiple generations. The GA in our model operates by encoding task allocation strategies as chromosomes, with fitness functions that evaluate the QoS performance of each strategy based on latency, energy consumption, and response time.

To ensure that no single QoS metric is overly prioritized at the

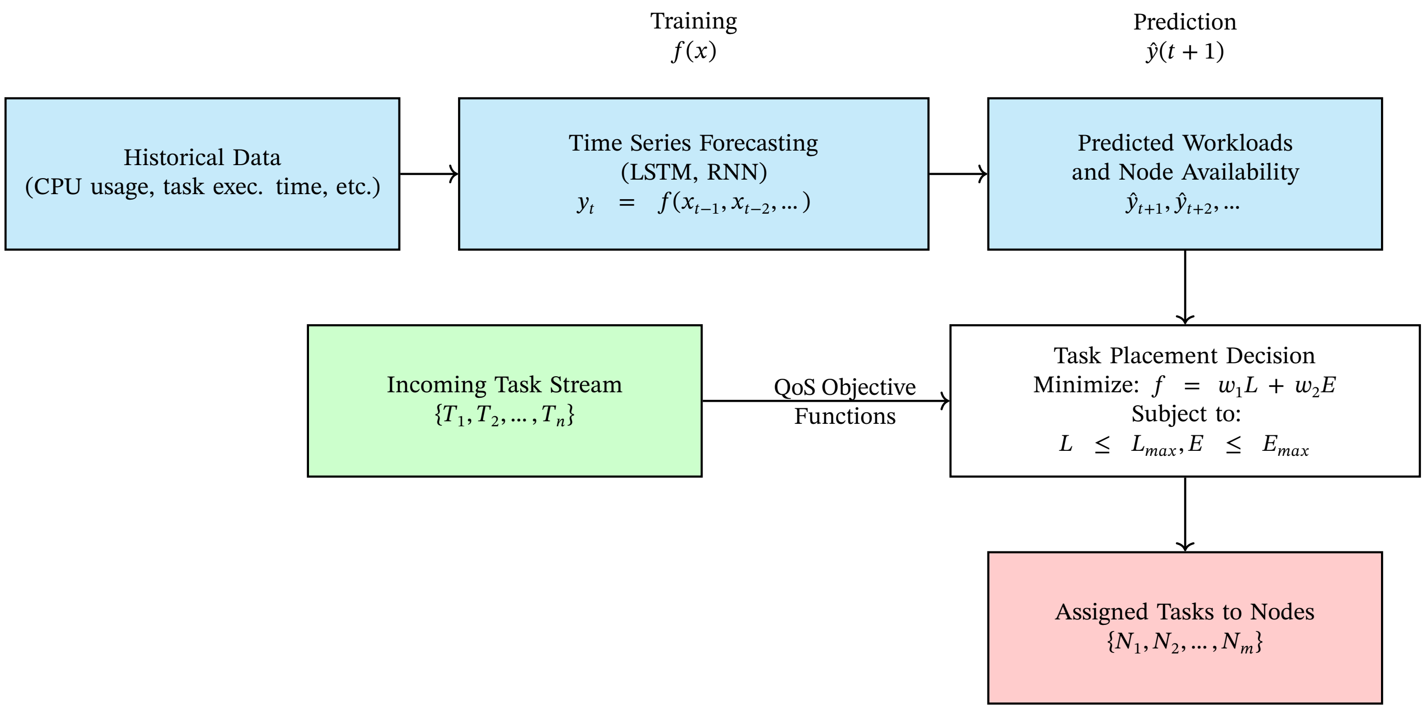


Figure 3. Architecture Diagram for Predictive Task Placement Module

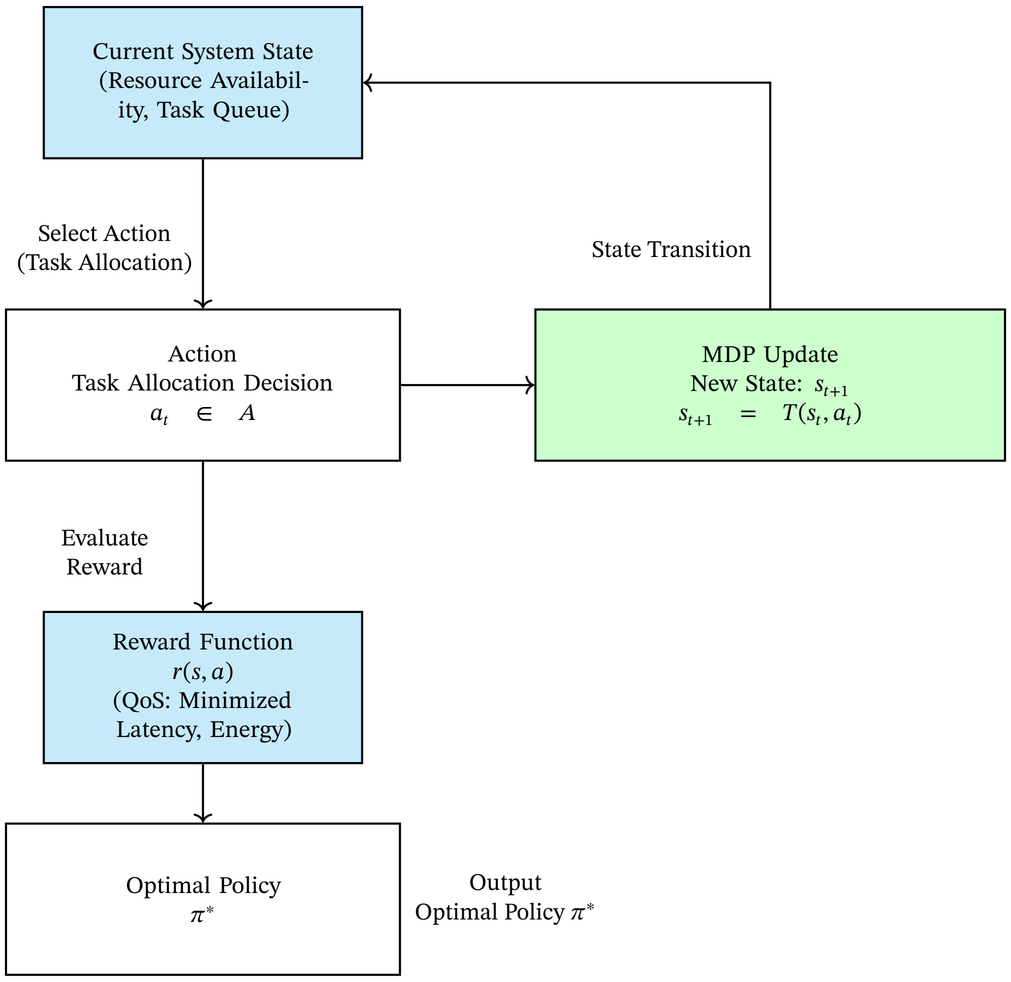


Figure 4. Architecture Diagram for Dynamic Adaptation through Reinforcement Learning

expense of others, we employ a Pareto optimization framework. In this framework, the system evaluates task allocation strategies based on Pareto dominance, where a solution is considered optimal if no other solution improves one QoS metric without degrading another. This approach allows us to generate a Pareto front of non-dominated solutions, from which the most suitable task allocation strategy can

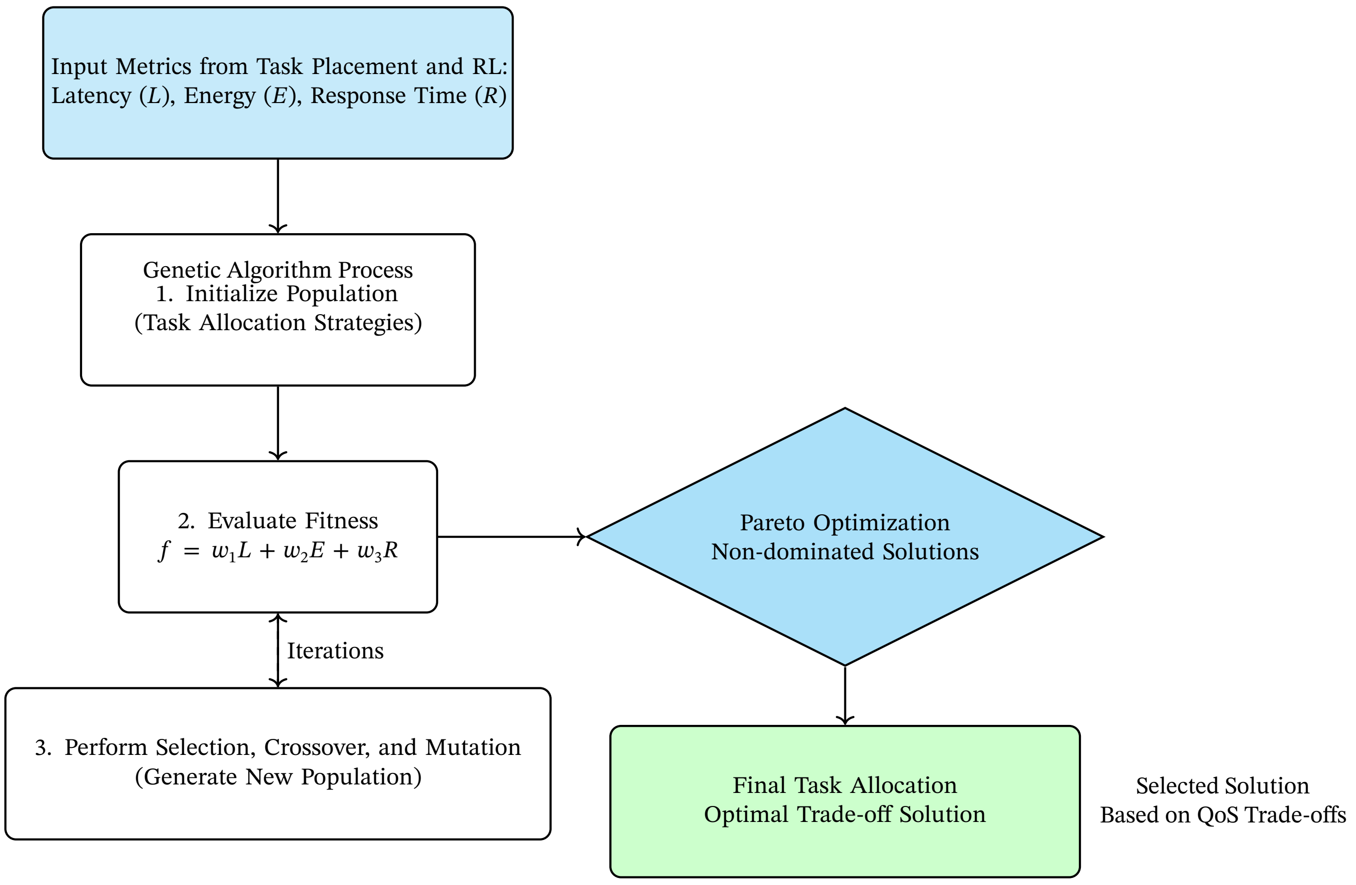

Input Metrics from Task Placement and RL:
Latency (L), Energy (E), Response Time (R)
Genetic Algorithm Process
1. Initialize Population
(Task Allocation Strategies)
2. Evaluate Fitness
$f = w_1L + w_2E + w_3R$
Iterations
3. Perform Selection, Crossover, and Mutation
(Generate New Population)
Pareto Optimization
Non-dominated Solutions
Final Task Allocation
Optimal Trade-off Solution
Selected Solution
Based on QoS Trade-offs

Figure 5. Multi-Objective Optimization using Genetic Algorithm and Pareto Front

be selected based on the current system requirements and priorities.

The hybrid nature of our multi-objective optimization approach allows the system to dynamically adjust the trade-offs between competing QoS metrics, ensuring that the system remains flexible and can adapt to changing conditions. For example, in situations where energy consumption is critical, the system may prioritize energy efficiency while still maintaining acceptable latency and response times. Conversely, during periods of high task demand, the system may prioritize minimizing latency to ensure timely task execution.

The modular nature of the proposed model allows for future enhancements, such as incorporating additional QoS metrics or extending the model to support larger-scale fog and edge computing networks. The integration of AI techniques ensures that the model remains adaptable to the dynamic and heterogeneous nature of fog computing, providing a scalable and flexible solution for real-time, latency-sensitive applications.

6. Conclusion

Fog computing environments are inherently heterogeneous, due to the diversity in fog nodes, which differ in processing power, storage, energy efficiency, and network bandwidth. Some nodes may have limited energy resources, such as battery-powered devices, while others offer high processing capabilities but may be geographically distant from the edge. This diversity presents significant challenges for resource allocation and task scheduling. Traditional scheduling models that assume uniformity across nodes often fail to optimize performance within such varied infrastructures. Artificial intelligence (AI) has become an essential component for optimizing task scheduling within heterogeneous environments like fog computing. Traditional scheduling algorithms, often based on heuristics, struggle to cope with the complexity and dynamic nature of fog systems. AI-based methods, especially those leveraging machine learning (ML) and reinforcement learning (RL), offer solutions that can learn from historical data and adapt to changing conditions. These techniques can predict future task demands, select optimal fog nodes for task execution, and adjust task assignments in real-time to enhance system performance. Task scheduling plays a critical role in the operation of fog computing systems. Effective scheduling ensures that tasks are assigned to appropriate fog nodes based on factors such as their computational capacity, energy consumption, and network proximity. The goal of task scheduling is to minimize system costs—such as latency, energy consumption, and response time—while ensuring that application-specific quality of service (QoS) requirements are satisfied. Unlike the cloud, where computational resources tend to be abundant and uniform, the constraints in fog environments demand more sophisticated, adaptive scheduling strategies.

Heuristic-based AI algorithms were among the earliest approaches applied to task scheduling in fog computing. These algorithms rely on predefined rules or criteria to make decisions, with AI techniques used to improve the selection and optimization of these rules. Examples include genetic algorithms, simulated annealing, and particle swarm optimization. Though less adaptive than ML or RL techniques, heuristic approaches strike a balance between complexity and performance, making them suitable for smaller or less dynamic fog environments.

Deep learning (DL) models convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are effective in modeling the complexities of task scheduling. These models can capture intricate relationships between task requirements and fog node capabilities, improving QoS metrics over time. DL-based algorithms are especially useful in large-scale data environments, where traditional models might struggle to handle the volume and variability of data flows.

Reinforcement learning (RL) has shown considerable promise in multi-objective optimization, where multiple QoS metrics must be optimized simultaneously. In fog computing, RL agents can be trained to balance key metrics—such as latency, energy consumption, and response time—while adapting to changes in network conditions and fog node availability. Multi-agent RL systems, where multiple agents collaborate to optimize different aspects of the system, can enhance scalability and performance in large fog environments.

Hybrid AI techniques combine the strengths of various AI methods to improve task scheduling. For instance, a hybrid approach might involve using machine learning to predict task loads, while reinforcement learning is employed to adapt scheduling decisions in real time. Another hybrid model could integrate deep learning with genetic algorithms to strike a balance between exploration and exploitation during scheduling. By leveraging the advantages of multiple AI techniques, hybrid approaches offer improved performance and adaptability.

This paper also presents an AI-driven task scheduling model designed to optimize multiple QoS metrics in heterogeneous fog computing environments. The model combines machine learning for predictive scheduling, reinforcement learning for real-time dynamic adaptation, and a hybrid approach for multi-objective optimization. Machine learning algorithms predict future task loads and node availability, ensuring efficient task allocation to minimize latency and energy consumption. Reinforcement learning agents continuously adjust task assignments in real time, adapting to changing network conditions to improve response times and avoid node overload. The hybrid AI framework balances latency, energy use, and response time, optimizing all QoS metrics without compromising system performance.

While the proposed AI-driven task scheduling framework demonstrates significant improvements in optimizing multiple QoS metrics in heterogeneous fog computing environments, it is not without limitations. Several challenges and areas for potential improvement exist, which could impact the performance, scalability, and practicality of the system under certain conditions.

One major limitation of the proposed framework is its reliance on accurate and extensive historical data for training the machine learning models in the predictive task placement module. In real-world fog computing environments, such historical data may be incomplete, noisy, or unavailable, especially in newly deployed systems or rapidly changing environments. Without sufficient training data, the accuracy of predictions regarding future task loads and node availability may suffer, leading to suboptimal task placements that negatively affect latency, energy consumption, and overall system performance. In addition, the model's ability to generalize to unseen or rare conditions, such as unexpected spikes in task demand or sudden node failures, may be limited if these scenarios were not adequately represented in the training data.

Another limitation is the complexity and computational overhead associated with integrating multiple AI-based components the reinforcement learning agents used for dynamic adaptation. While reinforcement learning provides a powerful mechanism for continuously adapting task scheduling decisions in response to changing network conditions, it can be computationally intensive during the learning phase. This could result in increased processing time and resource usage, which may be undesirable in resource-constrained fog environments where real-time responsiveness is critical. The exploration-exploitation trade-off inherent in reinforcement learning can lead to suboptimal decisions during the early phases of deployment, as the system requires time to learn optimal policies through interactions with the environment.

The multi-objective optimization module, while effective in balancing competing QoS metrics, also introduces challenges in terms of complexity and convergence. Genetic algorithms, which form the core of the optimization process, are known for their ability to explore large solution spaces; however, they may require significant computational resources and time to converge to a near-optimal solution in large-scale fog networks with many nodes and tasks. In scenarios where real-time decision-making is required, the time taken to perform multi-objective optimization could introduce delays that degrade system performance. Additionally, the Pareto-based optimization approach, while providing a balanced trade-off between QoS metrics, may not always yield solutions that are optimal for specific application requirements when the priorities of the QoS metrics shift dynamically over time.

As the number of edge nodes and tasks in a fog computing network increases, the computational complexity of both the predictive task placement and multi-objective optimization modules grows accordingly. In large-scale networks, the overhead associated with continuously predicting task loads, adapting to changing conditions, and optimizing multiple QoS metrics could become prohibitive. This may limit the applicability of the framework in environments with high task throughput or those requiring fine-grained control over task scheduling in real time. Techniques such as distributed optimization or hierarchical scheduling may be needed to address scalability issues, but these come with their own set of trade-offs, such as increased coordination overhead or reduced control over individual nodes.

Another limitation of the proposed framework is its assumption of accurate monitoring and communication capabilities across all nodes in the fog computing network. The framework relies heavily on real-time data from sensors and monitoring tools to inform its predictions, dynamic adaptations, and optimization decisions. In environments where communication is unreliable, such as in remote or highly congested networks, the quality of the data feeding into the scheduling framework could be compromised. This may result in inaccurate predictions, delayed responses, or suboptimal task allocations in scenarios where latency-sensitive applications are being deployed. Addressing this issue would require mechanisms to handle missing or delayed data, such as through redundancy, predictive analytics for missing data, or decentralized control systems.

The proposed framework primarily focuses on three QoS metrics: latency, energy consumption, and response time. While these are critical in most fog computing applications, there are other important QoS metrics that may be relevant depending on the specific use case. For example, metrics such as security, privacy, reliability, and fault tolerance are becoming increasingly important in modern fog and edge computing environments in healthcare, autonomous vehicles, and smart grids. The current framework does not directly account for these additional metrics, and extending the model to incorporate them would require significant modifications to both the optimization algorithms and the underlying architecture. This could introduce additional complexity and trade-offs, such as increased overhead or reduced focus on latency and energy efficiency.

The framework assumes that all nodes in the fog computing environment are equally capable of running machine learning models and reinforcement learning agents, which may not be practical in real-world scenarios. Fog networks often consist of highly heterogeneous devices, ranging from resource-rich servers to resource-constrained edge devices such as sensors and actuators. The computational demands of the AI-driven modules those involving reinforcement learning and multi-objective optimization, may not be feasible for all nodes in the network. This could necessitate the offloading of computationally intensive tasks to more powerful nodes or centralized cloud servers, introducing additional communication delays and negating some of the benefits of fog computing, such as low latency and localized processing.

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[2] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 46–52, 2018.

[3] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*, IEEE, 2015, pp. 73–78.

[4] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *big data and cognitive computing*, vol. 2, no. 2, p. 10, 2018.

[5] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM computer communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[6] F. Bonomi, R. Milito, J. Zhu, and S. 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*, 2012, pp. 13–16.

[7] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[8] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 federated conference on computer science and information systems*, IEEE, 2014, pp. 1–8.

[9] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things*, Elsevier, 2016, pp. 61–75.

[10] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[11] Y. Shi, G. Ding, H. Wang, H. E. Roman, and S. Lu, "The fog computing service for healthcare," in *2015 2nd International symposium on future information and communication technologies for ubiquitous healthCare (Ubi-HealthTech)*, IEEE, 2015, pp. 1–5.

[12] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[13] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare–a review and discussion," *IEEE Access*, vol. 5, pp. 9206–9222, 2017.

[14] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[15] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, pp. 25 445–25 454, 2017.

[16] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," *Internet of everything: algorithms, methodologies, technologies and perspectives*, pp. 103–130, 2018.

[17] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.