UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Jeyhun Abbasov

AI-powered cloud usage controlling system in smart home environment

Master's Thesis (30 ECTS)

Supervisor: Chinmaya Kumar Dehury, Ph.D.

Tartu 2023

AI-powered cloud usage controlling system in smart home environment

Abstract: Cloud computing is utilized for handling and keeping the large amounts of data produced by IoT devices in a smart home environment. One of the major limitations of cloud computing is network latency. Due to these limitations, the Fog computing is introduced. Fog computing provides real-time services and saves network resources. A large portion of research are aimed at ensuring high-quality service delivery through the utilization of either fog or cloud environments. In our study, we present a Deep Reinforcement Learning service distribution solution using both fog and cloud computing environments in real-time manner. The demand for services is divided and distributed among the fog and cloud environments without compromising the Quality of Services (QoS). The results of implementation demonstrate substantial enhancement compared to state of the art studies.

Keywords: Cloud Computing, Fog Computing, Deep Reinforcement Learning, Service Delivery, Service Dispersal

CERCS: P170 Computer science, numerical analysis, systems, control

AI-jõuline pilvekasutuse juhtimissüsteem nutikas kodu keskkonnas

Lühikokkuvõte: Pilvandmetöötlust kasutatakse IoT-seadmete toodetud suurte andmemahtude haldamiseks ja hoidmiseks nutikas kodukeskkonnas. Üks pilvandmetöötluse peamisi piiranguid on võrgu latentsus. Nende piirangute tõttu võetakse kasutusele uduarvutus. Uduarvutus pakub reaalajas teenuseid ja säästab võrguressursse. Suur osa uuringutest on suunatud kvaliteetse teenuse osutamise tagamisele kas udu- või pilvekeskkonna kasutamise kaudu. Oma uuringus tutvustame Deep Reinforcement Learning teenuse levitamislahendust, mis kasutab reaalajas nii udu- kui ka pilvandmetöötluskeskkondi. Nõudlus teenuste järele on jagatud ja jaotatud udu- ja pilvekeskkondade vahel, ilma et see kahjustaks teenuste kvaliteeti (QoS). Rakendamise tulemused näitavad olulist paranemist võrreldes nüüdisaegsete uuringutega.

Võtmesõnad: Pilvandmetöötlus, Uduarvutus, Sügav Õpetamisreinforcement, Teenuse Tarnimine, Teenuse Jaotus

CERCS:P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Acknowledgements

I would like to thank my family, friends who always motivated me during this research and my supervisor Prof. Dr. Chinmaya Dehury for giving me detailed guidance, providing direction to the correct path.

Contents

1 Introduction

With the rise of the Internet of Things (IoT), smart homes have become a reality, offering greater convenience and control over our daily lives [Ela19]. IoT devices are equipped with sensors and actuators that allow us to monitor and control various aspects of our home environment, such as temperature, lighting, and security, from anywhere at any time through dedicated mobile applications or web services. This has revolutionized the way we live, work, and interact with our homes. In addition, the increasing demand for computing and storage resources from these smart home services is driving the development of cloud computing infrastructure, making it possible to store and process large amounts of data in real-time, further enhancing the capabilities of our smart homes.

Cloud computing is comprised of a vast variety of powerful physical servers that provide an almost infinite number of computation, storage, and networking capabilities to enable IoT services. In cloud computing, there are typically three categories of services: infrastructure as a service (Iaas), platform as a service (Paas), and software as a service (Saas). IaaS is the most basic type of cloud computing. It provides access to a virtualized environment that includes computer resources such as servers and virtual machines, storage, networks, and operating systems. PaaS provides an on demand platform for developing, testing, deploying, and managing software without having to worry about establishing or maintaining the necessary IT infrastructure such as servers, storage, networking, and databases. SaaS is a vendor-provided cloud-based software that may be accessed via a web browser. Customers may subscribe to it and enjoy a variety of applications like as accounting, sales, invoicing, and more. The advantages of SaaS include low cost, great performance, and dependbility. Despites its advantages, cloud computing has drawbacks, such as geographical constraints and sophisticated network architecture [SDV18]. This is where Fog Computing comes in, filling the gap between the traditional cloud computing and IoT devices. It extends cloud computing closer to the edge devices, improving mobility support, enabling location awareness, and reducing latency for better performance and user experience. [SWHL16]

Fog computing is proposed as a solution to address the previously mentioned issue. Fog computing solves the issues of low latency, location awareness and improves quality-of-service for real time and streaming applications by being deployed at the edge of the network. Examples of fog computing applications can be found in industrial automation, transportation and networks that involve sensors and actuators. In addition, the fog infrastructure accommodates different types of devices, such as user end devices, access points, edge routers, and switches. Fog computing is ideal for real-time big data analytics and provides benefits in various fields like entertainment, advertising, personal computing, etc. by enabling densely distributed data collection points. By utilizing local resources in fog computing, a user's service requirements can be met with less reliance on distant cloud resources, leading to improved performance. Implementing fog computing can help reduce service latency, improve overall Service Response Time (SRT), reduce

network congestion, decrease energy consumption, and enhance system security by utilizing resources at the edge organized as fog nodes [SMBMT+18, YHQL15].

Edge computing is an extension of Fog Computing where data is processed closer to the source, either on the device, sensor, or a nearby server. One of the crucial elements in an edge computing is the Smart Gateway. The Smart Gateway is responsible for managing various IoT operations, such as collecting and preprocessing data, filtering and reformatting it, uploading relevant data to the fog/cloud, monitoring IoT objects and sensors, monitoring energy consumption, ensuring data security and privacy, and overall service management. Additionally, The Smart Gateway can play a key role in efficient service dispersal process. Communication between IoT and Smart Gateway can either be direct or through base station(s). There are two types (Single-hop communication, Multi-hop communication) of Smart Gateway-based communication. In single-hop connectivity, the sensors and IoT devices are directly connected to the Smart Gateway. The Smart Gateway then collects the data and sends it to either the Fog or Cloud. This type of communication is typically used in smaller networks where the sensor nodes have limited roles, such as in smart health or healthcare applications. Quick monitoring and response is possible through direct communication with the Smart Gateway. M2M communication takes place in this scenario, and the Smart Gateway can perform data refinement, filtering, trimming, and security measures, service dispersal process, depending on the application needs and in conjunction with fog computing. The extent of this type of communication depends on the capabilities of the Smart Gateway device. In a scenario where multiple sensor networks and IoTs are connected, direct connections become unfeasible. These networks and IoTs have their own base stations and sink nodes, and the gateway collects data from these. This creates a multi-hop communication setup, where nodes are more diverse and spread out and data is more heterogeneous, requiring more processing and analysis from the gateway. Sink nodes provide an additional layer of communication and make underlying sensors and things a "black box," adding security. Security can be customized according to the IoT and WSN, and sink nodes can manage sensor networks based on their constraints. The gateway must handle heterogeneous data from various devices, IoTs, and WSNs, requiring transcoding and interoperability. This can either be achieved through an intelligent gateway or through Fog computing resources. This setup is suitable for large scale IoTs/WSNs and mobile objects, such as vehicle tracking IoTs and environmental monitoring. [AH14] For the purpose of keeping things simple, we have selected to use Single-hop communication in our Smart Gateway setup. As an illustrative example, Figure 1 illustrates an abstract view of Smart Gateway with Cloud-Fog-Edge architecture.

In our approach, the service workload is divided into smaller sub-services by Smart Gateway and is shared between fog and cloud servers. The fog node processes some slices of the service request and the rest is sending to the cloud for processing. The decision of which slices to be processed by the fog and cloud depends on various factors such as user

proximity, user priority, service sensitivity, energy consumption, etc. Determining the correct distribution of the service between fog and cloud becomes increasingly complex as the service complexity grows, leading to a new research challenge.

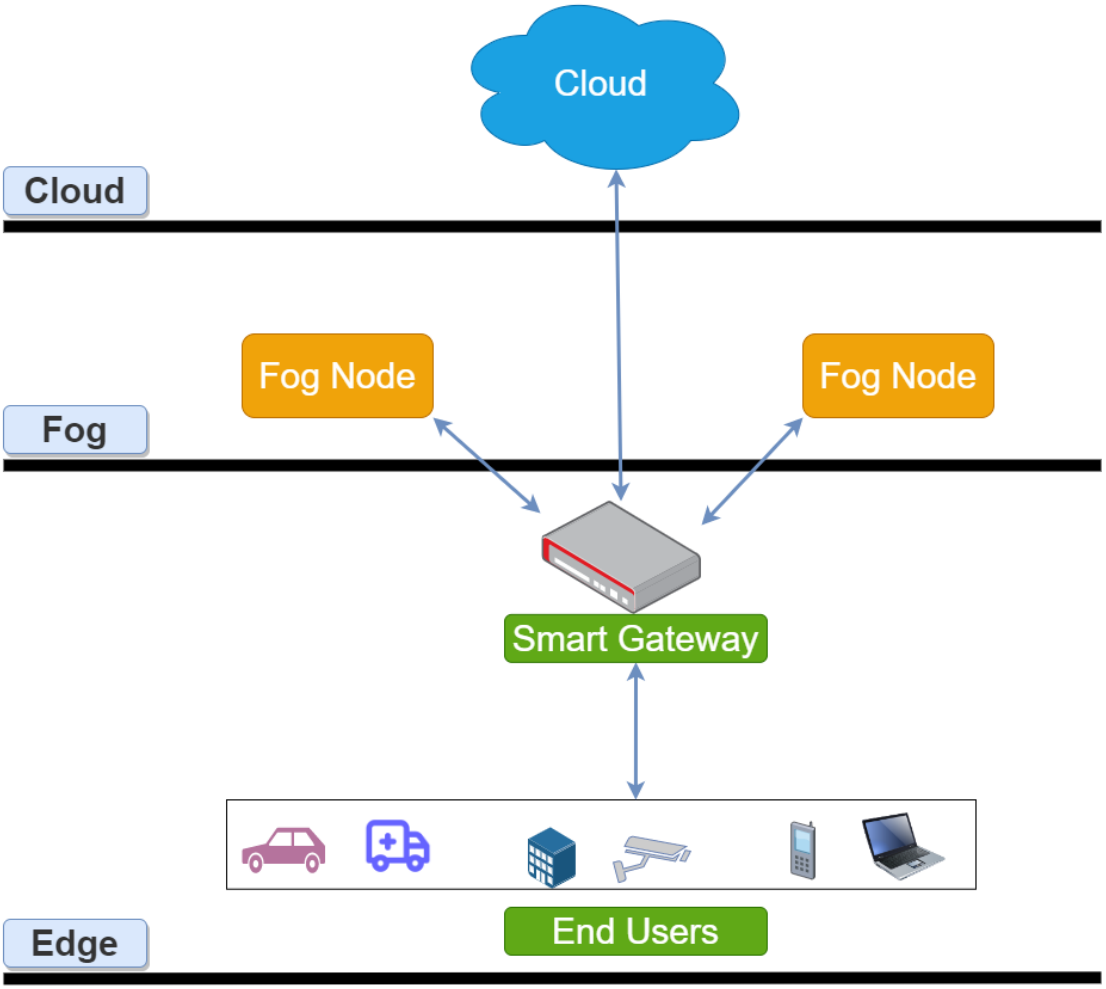

Figure 1. Abstract view of Smart Gateway with Cloud-Fog-Edge architecture.

1.1 Motivation

The allocation of services between the fog and cloud is managed by the Smart Gateway, as previously discussed, to provide services to the users through both fog and cloud environments. As seen in Figure 2, the division of services between fog and cloud for different services and users is not consistent, and an individual user may utilize a variety

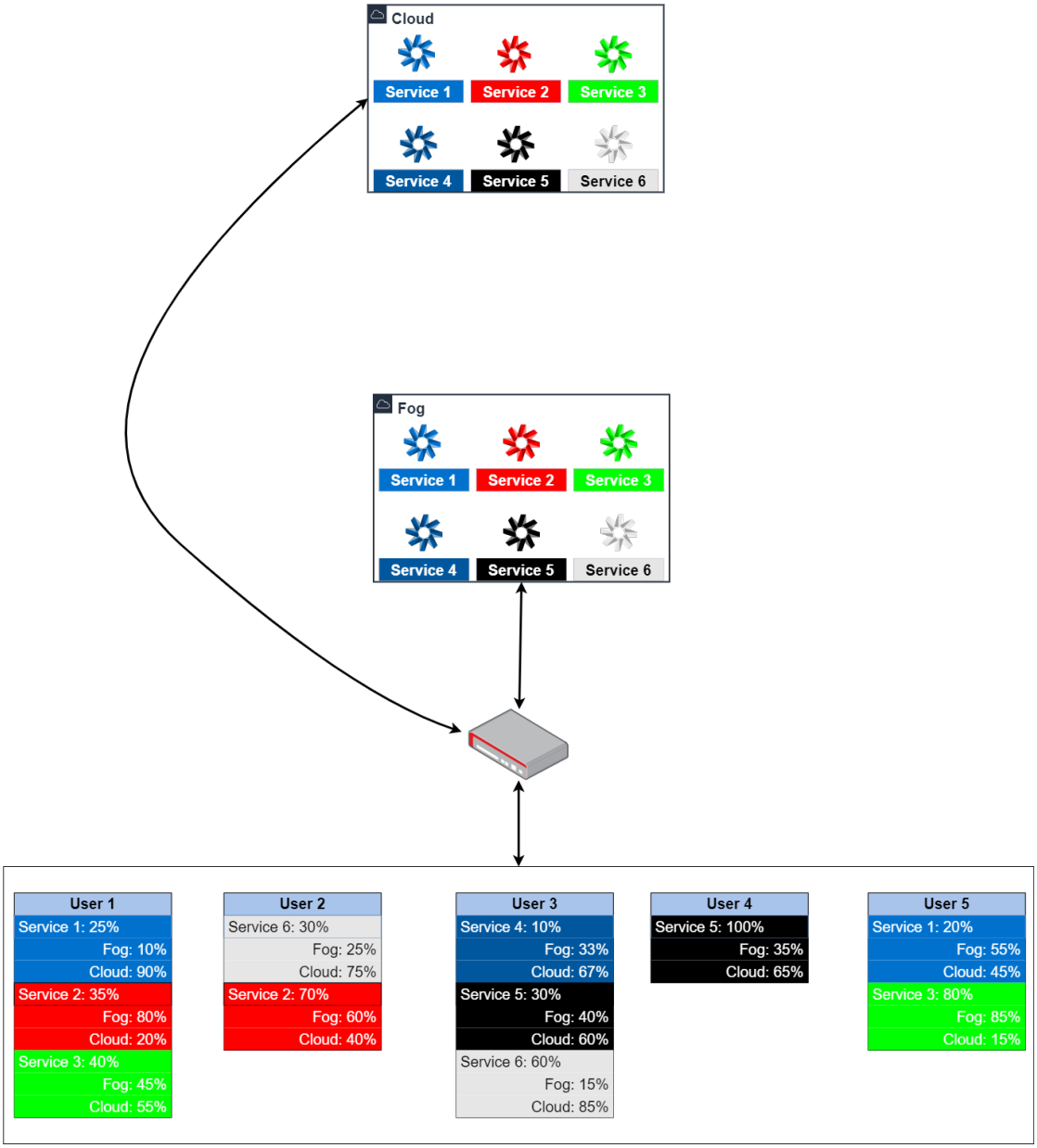

Figure 2. An example motivational scenario.

of services. For example, the User1 is using three services named as Service1, Service2, and Service3. The service request from User1 includes a request for 25% of Service1, a request for 35% of Service2, and a request for 40% of Service3. The 45% and 55% of Service3 that is requested by User1 is handled by Fog and Cloud, respectively. The distribution of service workload between fog and cloud servers can vary, depending on

various factors such as the sensitivity of the service, user location, available resources, and others. For example, a critical service with low tolerance for latency cannot be assigned to the cloud, whereas a fog node experiencing resource constraints may need to delegate tasks to the cloud. Balancing the workload between fog and cloud environments becomes challenging when taking into account multiple parameters. This motivates us to investigate a new solution for distributing service workload effectively between fog and cloud computing platforms while maintaining quality of service. In this thesis, to overcome the aforementioned issue, we are adopting deep reinforcement learning.

1.2 Goal and Contributions

The goals of this paper are to enhance QoS with reduced service latency by delegating a portion of the service to the cloud, serve the maximum number of users in real-time by letting the cloud handle part of the service request, and efficiently utilize the fog node's limited resources by incorporating the cloud in the real-time service delivery process. The main contribution of this work can be summarized as follows:

- This paper outlines the problem of distributing users' services among fog and cloud environments;
- Proposed a new deep reinforcement learning (DRL) based method to distribute service requests to fog and cloud computing environments;
- The proposed DRL approach performance is evaluated in terms of metrics such as success rate, distribution of service requests, and etc.
- The results are compared with existing state-of-the-art methods.

1.3 Outline

This chapter provides an overview of IoT, cloud computing, fog computing, smart gateway, the significance of service distribution, and how deep reinforcement learning can help resolve the issue. Chapter 2 delves into the background of service distribution and reviews the current state of the art. In Chapter 3, we present our solution to the service distribution problem by discussing the architecture, applications, methods, and algorithms used. Chapter 4 details the experiment process and results are discussed in Chapter 5. Finally, Chapter 6 concludes the work with a summary and future direction.

2 State of the Art

In this section, we delve into the current state of the art in the field of service delivery. Our discussion begins by providing an overview of the background of service delivery in both fog and cloud computing environments in Section 2.1. This section aims to give the reader a comprehensive understanding of the context in which the current work is being carried out. Following this, in Section 2.2, we delve into the related works that have been done in the area of service dispersal methods within the cloud domain. This section aims to give the reader an understanding of the prior efforts-made in this field and the gaps that the current work aims to fill. Finally, in Section 2.3, we provide a summary of the key points covered in this section, offering a clear and concise overview of the state of the art in the field of service delivery.

2.1 Background

2.1.1 Cloud Computing

Recently, the Internet of Things (IoT) has made a significant impact on our society by converting everyday objects like wearables, transportation, and augmented reality into communicating devices, bringing both new challenges and opportunities. It is evident that the existing infrastructure will not be capable of handling the vast amount of data generated as the number of devices connected to the network is expected to reach over 50 billion by 2020, according to Cisco [Kim16]. The current Cloud infrastructure is inadequate to support a substantial amount of the existing IoT applications, due to three main reasons. First, the large volume of generated data makes it difficult to transfer it from the end-devices where it is created to the Cloud servers where it is processed. This is due to limitations in bandwidth, processing overhead, and transmission costs. Second, applications that require real-time analysis, such as online gaming and video applications, can be negatively impacted by the significant end-to-end delay from end-devices to the faraway Cloud servers. Lastly, privacy and security concerns may dictate that certain sensitive data must not traverse the entire Internet, making it necessary to process it closer to its source [YQL15].

A solution has been identified to address the problems mentioned above by avoiding network congestions, reducing communication costs, and minimizing the delay in data transfer. This new concept that leverages the advantages of the Cloud and the decentralization of service processing on edge devices is called Fog Computing [BMNZ14].

2.1.2 Fog Computing

Fog Computing extends Cloud Computing by locating processing, analysis, and storage closer to the source of requests, at the edge of the network. The goal is to minimize the

data sent to the Cloud and lower the time delay and computational expenses [DGC+16]. One of the main challenges of Fog Computing, a new paradigm, is the management of services, specifically the problem of determining the appropriate placement for them. Efficently deploying services on available Fog nodes is a significant challenge facing the widespread adoption of Fog Computing. The difficulty in deploying services on Fog nodes arises from the fact that they are geographically dispersed, have limited resources, and can change dynamically, adding to the complexity of the issue. Therefore, it is crucial to determine an efficient, effective, and equitable way of allocating resources to IoT applications in order to provide end-to-end guaranteed services to the users.

Additionally, the focus may vary depending on the situation and the priorities, including aspects such as resource utilization [TD17], Quality of Service (QoS) [SNSD17, BF17], and Quality of Experience (QoE) [MSRB19]. Over the past few years, multiple efforts have been made to address this challenge. An efficient service placement has been proposed by taking into account different characteristics, assumptions, and strategies.

2.1.3 Edge Computing

Edge computing extends Fog Computing, and refers to the practice of processing data near the edge of the network, closer to the source of the data. In this paradigm, processing takes place on the device or sensor itself or on a nearby server. The goal of edge computing is to reduce the amount of data that needs to be transmitted to the fog or cloud for processing, which reduces latency and improves performance. One of the main benefits of edge computing is its ability to support mission-critical applications that require low latency and high reliability. For example, edge computing can be used in the healthcare industry to provide real-time monitoring and analysis of patient data, enabling medical professionals to make informed decisions quickly.Edge computing architectures can vary, with some relying on edge devices like smartphones, and others using dedicated edge servers. [SDL20]

Overall, by bringing processing and storage capabilities closer to the source of data, edge computing enables new applications and services that were not possible before, while also providing faster and more reliable processing and analysis of data.

2.2 Related Work

The placement of services that are effective in fog and cloud environments is a complex task that is influnced by several factors. The first of these is the heterogeneous and resource-constrained nature of most Fog nodes, which have limited capacities. The second factor is the dynamic nature of the environment, where resources can change over time and workload can be vary. Thirdly, the issue is made even more complicated by the widespread distribution of fog devices across a large-scale infrastructure. The problem of SPP (service placement problem) in a fog and cloud environment is a difficult

task due to various specificities and constraints. Within the literature, we can identify the primary approaches employed to solve this problem as heruistic approach, machine learning based algorithms, and deep reinforcement learning based algorithms.

2.2.1 Heruistic Approaches

Wang and Li et. al [WL19] proposes a solution for improving task scheduling in fog computing using a hybrid heuristic algorithm. However, this study is limited to only using bandwith and energy consumption.

Zahoor et al. [ZJJ+18] proposes a model for managing resources in SGs using cloud-fog computing, and uses several load balancing algorithms including HABACO to optimize response time. This study is not using the deep reinforcement learning approach as our study.

Rafique et al. [RSI+19] proposes a scheduling strategy to optimize resource utilization and reduce average response time in fog computing. The proposed approach outperforms other techniques in terms of energy consumption and execution time. However, it is required to apply deep reinforcement learning techniques to improve fog-IoT resource management.

Yasmeen et al. [YJR+18] proposes a three-layered cloud and fog-based model to decrease consumer load and power generation system. Resource balancing is achieved through the use of three algorithms: Round Robin, throttled, and Particle Swarm Optimization with Simulated Annealing (PSOSA). Unlike our study, this study does not take reducing latency into account.

Yadav et al. [YNG19] proposes a hybrid algorithm that combines Genetic Algorithm and Particle Swarm Optimization to allocate services efficiently in a fog computing environment. The algorithm minimizes total makespan and energy consumption for IoT applications. Ren et al. [RZA21] suggests a model to decrease energy consumption and efficiently manage energy resources in fog systems. The proposed architecture aims to manage resources in a way that reduces losses and ensures quality of service. However, both these studies is only based on energy consumption unlike ours.

Hosseinioun et al. [HKTG20] suggests a method for saving energy in the fog computing environment using the Dynamic Voltage and Frequency Scaling (DVFS) technique. A hybrid algorithm, IWO-CA, is used to create valid task sequences.

Javanmardi et al. [JSPP21] The article describes the FPFTS fog task scheduler, which uses particle swarm optimization and fuzzy theory to optimize application loop delay and network utilization. The drawback of the study is that it requires developing a mobility-aware task scheduling algorithm. The study clearly lacks the benefit of other features like resource usage as in our study.

Xu et al. [XHZS19] proposes the LBP-ACS algorithm for task scheduling in cloud-fog environments, which considers priority constraints, energy consumption, and mixed deadlines. The algorithm reduces energy consumption and failure rates while ensuring

reasonable scheduling length. The study is limited to using only energy consumption and deadlines.

Djemai et al. [DSMP19] proposes a strategy for IoT services placement in Fog architecture using a Discrete Particles Swarm Optimization algorithm. The placement strategy minimizes application delay violations and considers energy consumption. This study only considers minimizing energy consumption.

2.2.2 Machine Learning based Algorithms

Rahbari and Nickray et al. [RN20] presents a Module Placement method by Classification and regression tree Algorithm (MPCA) for selecting the best FDs for modules. MPCA selects the best FDs based on decision parameters such as authentication, confidentiality, integrity, availability, capacity, speed, and cost. As opposed to our study, these studies are focusing on the module placement, there is no slicing method for modules.

Bashir et al. [BLK22] proposes a dynamic resource allocation strategy for cloud, fog nodes, and users. The strategy uses TOPSIS (technique for order performance by similarity to ideal solution) to rank the fog nodes and logistic regression to calculate the load of individual fog nodes. This study though uses the machine learning approach for dynamic resource allocation but is limited to slicing.

Ferrández-Pastor et al. [FPMJMV18] proposes a method to design smart services based on the edge computing paradigm. The approach addresses interoperability and scalability issues of existing designs and implements energy management, security system, climate control, and information services subsystems on embedded devices. This study, though, makes use of the machine learning approach but fails to minimizing resource usage like our study.

He et al. [HWC+17] proposes a new fog computing model with functional modules can reduce problems related to dedicated computing infrastructure and slow response times in cloud computing. However, this study is limited to only using response time.

Priyabhashana et al. [PJ19] compares the use of various activation functions on a TensorFlow-based neural network model built using the Keras library. The proposed fog station was evaluated, and the experiment results showed the feasibility, efficiency, and applicability of the system. This study is focused on cpu and ram usage.

Alsaffar et al. [APH+16] proposes an architecture for IoT service delegation and resource allocation that combines fog and cloud computing. The authors propose an algorithm to allocate resources to meet SLA and QoS requirements and optimize big data distribution. However, this study is not considered using bandwith and energy consumption.

Yadav et al. [YMY20] presents a new technique for task allocation that minimizes response time and system cost using clustering. The proposed technique uses Fuzzy C-Means clustering and Hungarian method for task allocations. However, this study is only based on minimizing response time unlike ours.

2.2.3 Deep Reinforcement Learning based Algorithms

Farhat et al. [FSM20] proposes R-learning model that aims to decrease the cloud's load by utilizing available fog resources in different locations. The model is demonstrating its ability to adapt to user demand and efficiently use fog resources. However, this study is limited to only using user demands.

Orhean et al. [OPR18] presents a reinforcement learning algorithm for solving the scheduling problem in distributed systems. The proposed algorithm considers the heterogeneity of nodes and the dependency graph of tasks for determining a scheduling policy that minimizes execution time. Unlike our study, this study only take execution time into account.

Tang et al. [TZZ+18] proposes a new architecture and algorithms for container migration to support mobility tasks with different application requirements. The proposed algorithms consider hosting mobile application tasks in containers of corresponding fog nodes, and modeling container migration strategies as multiple dimensional Markov Decision Process spaces. This study though uses the deep reinforcement learning approach but is limited due to slicing.

Gazori et al. [GRN20] proposes a Double Deep Q-Learning (DDQL)-based scheduling algorithm for task scheduling of fog-based IoT applications. The aim is to minimize long-term service delay and computation cost under resource and deadline constraints. As opposed to our study, these studies are focusing on minimizing service delay, and deadline constraints.

Alam et al. [AHU+19] proposes a deep Q-learning based autonomic management framework for computation offloading in mobile edge/fog. The framework uses a distributed edge/fog network controller to allocate resources and minimize latency of service computing while maintaining energy efficiency. However, this study is only based on energy consumption and minimizing service latency unlike ours.

Zhang et al. [ZLY+18] proposes a double deep Q-learning model for energy-efficient edge scheduling (DDQ-EES). The model includes a generated network and a target network for producing Q-values, and rectified linear units (ReLU) as the activation function to avoid gradient vanishing. This study only considers energy-efficiency unlike our study.

Lu et al. [LHD+20] proposes a QoE model for computation offloading that considers service latency, energy consumption, and task success rate. The study clearly lacks the benefit of other features like resource usage as in our study.

Wang et al. [WLLC19] proposes a Q-learning-based hierarchical service tree placement strategy to optimize the net utility in large networks with complex service structures. The study is limited to using only network bandwidth feature.

Dehury and Srirama et al. [DS19, DS20] a reinforcement learning-based (RLPSD) and a deep reinforcement learning-based (DRLSD-FC) service delivery mechanism that distributes users' service requests between fog and cloud environments to to minimize

the workload on fog while maintaining service quality. The algorithms consider user constraints such as distance from fog, service sensitivity, and other metrics. However, the study is limited because it does not address service-placement in real-time manner, energy consumption and network bandwidth as our study, and service dispersal mechanism inside of fog nodes, in our study we use smart gateway in order to do service dispersal jobs.

Table 1, shows the summary of related works that are used in this paper. First column of the table shows name of paper, following columns represents features of the papers.

Table 1. Summary of Related Work

Paper Name	**User Priority**	**Service Priority**	**Slicing**	**Resource**	**Network**	**Energy**
[WL19]	-	-	-	-	-	-
[ZJJ+18]	-	-	-	-	-	-
[RSI+19]	-	-	-	-	-	-
[YJR+18]	-	-	-	-	-	-
[YNG19]	-	-	-	-	-	-
[RZA21]	-	-	-	-	-	-
[HKTG20]	-	-	-	-	-	-
[JSPP21]	-	-	-	-	-	-
[XHZS19]	-	-	-	-	-	-
[DSMP19]	-	-	-	-	-	-
[RN20]	-	-	-	-	-	-
[BLK22]	-	-	-	-	-	-
[FPMJMV18]	-	-	-	-	-	-
[HWC+17]	-	-	-	-	-	-
[PJ19]	-	-	-	-	-	-
[APH+16]	-	-	-	-	-	-
[YMY20]	-	-	-	-	-	-
[FSM20]	-	-	-	-	-	-
[OPR18]	-	-	-	-	-	-
[TZZ+18]	-	-	-	-	-	-
[GRN20]	-	-	-	-	-	-
[AHU+19]	-	-	-	-	-	-
[ZLY+18]	-	-	-	-	-	-
[LHD+20]	-	-	-	-	-	-
[WLLC19]	-	-	-	-	-	-
[DS19]	-	-	-	-	-	-
[DS20]	-	-	-	-	-	-

2.3 Summary

In this chapter, we first introduced with background information on service placement in fog and cloud environments. Then we discussed the current state-of-the-art in service dispersial methods for both environments. We also saw the different approaches used to answer the SP problem in related works section. Finally, we saw what is new in our studies compared to the current state-of-the-art. In the next chapter, we will see our approach to the problem of service placement in fog and cloud environments.

3 Methodology

Multiple methods can be used to solve the same issue. In this section, we outline our approach to addresing the challenge of distributing services between fog and cloud environment. We start our discussion by modeling the services and mathematical representation and analysis of key parameters in Section 3.1, followed by a deep reinforcement learning, covering topics like state space, action space, and reward calculation in Section 3.2. Section 3.3 delves into the selected applications and their architectures for our study. Finally, we wrap up this section with a summary in Section 3.4.

3.1 System Architecture

In Figure 2, the fog and cloud model offers multiple services to users. Users can obtain services by submitting a Service Request (SR) to the smart gateway. The Service Request can be partitioned, which is an assumption that can be demonstrated with various real world examples involving data intensive applications generated by IoT devices, distributed among the fog and cloud environments. The smart gateway collaborates with both fog and cloud environments to meet service resource demands.

3.1.1 Smart Gateway

The Smart Gateway, represented by G, plays a vital role in ensuring the efficient distribution of user requests within the fog and cloud computing environments. G's primary task is to break down user requests into smaller, manageable slices and allocate these slices across the fog and cloud environments. Each slice is a logical subdivision of the service request or input data, allowing for efficient processing and improved overall performance. For instance, if a user uploads a 250MB file, G can divide this file into five slices with varying sizes, each containing $\sim 50MB$ of the data. This way, the processing of the file is distributed and can be handled more efficiently.

To allocate these slices effectively, G considers several essential factors that influence the performance and user experience. These factors include user priority, service sensitivity, service priority, resource demand, energy consumption, and more. For example, service priority takes into account the importance of the request, with higher-priority tasks being assigned more resources; service sensitivity is another key factor, as some services may require more stringent privacy or security measures; lastly, energy consumption is considered to optimize the overall efficiency of the system, aiming to reduce energy usage where possible. By evaluating these factors, G is able to distribute the slices effectively, ensuring optimal performance and user satisfaction in the fog and cloud computing environments.

3.1.2 Fog and Cloud Environments

The Fog Node, represented by F, is a crucial component within the fog and cloud computing environment, outfitted with a limited amount of resources such as CPU and RAM. Despite these limitations, F is designed to offer a set of m number of services, denoted by $S = \{s_1, s_2, s_3, ..., s_m\}$, where each service is represented as s_k, with $0 < k \leq m$. The primary function of the fog node is to provide these services to the maximum number of connected users within its proximity, ensuring low-latency and efficient processing of user requests. By functioning as an intermediary between smart gateway and the cloud, the fog node helps to reduce the load on the cloud environment.

On the other hand, the cloud node, represented by C, is equipped with a vast array of interconnected high-end servers, providing virtually unlimited processing power and storage capacity. Unlike the fog node, the cloud node is designed to handle more resource-intensive tasks, supporting a multitude of users and services simultaneously. While the F is responsible for receiving and processing requests from smart gateway, some tasks are may delegated to the cloud node when required. This could occur when the fog node is unable to handle the demand or when specific tasks necessitate the capabilities of the cloud node. By utilizing the strengths of both the fog node and the cloud node, the fog and cloud computing environments can ensure optimal performance, efficient resource allocation, and a seamless user experience.

3.2 User Modelling

The fog and cloud computing environments offer a diverse range of services to cater to the various needs of all connected users. In our study, $U = \{u_1, u_2, u_3, ..., u_n\}$ represents the set of all connected users, where each user is denoted by u_i, with $0 < i \leq n$. Users have the flexibility to send requests for multiple service types, depending on their specific requirements. The boolean variable $u_i(s_k)$ is used to indicate if user u_i is currently availing the service s_k. $u_i(s_k)$ can be represented mathematically as following [DS20]:

$$u_i(s_k) = \begin{cases} 1 \text{ - if user } u_i \text{ is availing the service } s_k \\ 0 \text{ - otherwise} \end{cases} \tag{1}$$

This variable helps the Smart Gateway, or G, track the active services for each user, allowing for optimal service delivery. As users interact with the fog and cloud computing environments, two major parameters play a crucial role: distance priority and latency priority.

3.2.1 User Distance Priority

The geographic distance between a user and the G impacts the overall latency and performance of the system. As the distance increases, the time taken for data to travel

between the user and the gateway may also increase, potentially leading to a slower user experience. A user is positioned within the maximum range of the G, with each user and the G having associated location parameters. The distance between a user and the Smart Gateway can be computed using the Haversine formula as follows:

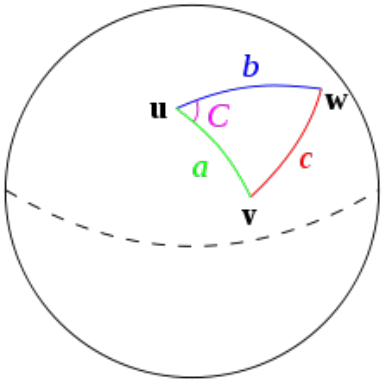

Figure 3. Haversine law

$$d_i = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{p_g - p_i}{2}\right) + \cos(p_i) \cdot \cos(p_g) \cdot \sin^2\left(\frac{q_g - q_i}{2}\right)}\right) \quad (2)$$

In this formula, r represents the radius of the sphere, (p_i, q_i) represents the latitude and longitude of user u_i, and (p_g, q_g) corresponds to the latitude and longitude of the Smart Gateway. Figure 3 illustrates a Haversine law. The distance of a user has to be less than the maximum range of the Smart Gateway [DS20]. The maximum range of the Smart Gateway is represented by D. By considering the D and the user's distance (d_i), the distance priority of a user can be determined as follows:

$$P_i^d = \frac{d_i}{D}, \forall u_i \in D \quad (3)$$

All users who are equidistant from the G will have the same priority based on distance, with those farther away given higher priority.

3.2.2 User Latency Priority

Communication latency refers to the duration required for a user's request to be sent to the Smart Gateway and for the subsequent response to be returned. This latency can be influenced by a variety of factors, such as network congestion or constraints in the user's device. Users closer to the Smart Gateway may not necessarily experience lower latency.

Let l_i represent the Round Trip Delay (RTD) time for user u_i. Based on this value, the latency-based priority can be calculated as:

$$P_i^l = \frac{l_i}{\max(l_i, 0 < i \leq n)} \tag{4}$$

The value of P_i^l is relative and depends on the latency values for all connected users. Users with higher values of P_i^l are assigned the highest priority.

3.2.3 User Priority

We explored the methods for calculating user distance and latency priorities in the previous subsections. Taking into account these priorities, as shown in Equation 3 and Equation 4, we can determine the combined priority of a user using the following approach:

$$P_i = P_i^d + P_i^l \tag{5}$$

By considering both distance and latency priorities, we can develop a more comprehensive understanding of each user's needs within the fog and cloud computing environment. This overall priority calculation allows for more effective service request allocation and ensures that users with more urgent or demanding requirements are given the necessary attention.

3.3 Service Modelling

Here, we delve into service modeling and its various aspects. When users send a service request, several properties of the services must be considered, such as the size of the service request and the demand for the service request, etc. A service request (SR) from a user is divided into $\hat{s}_i^k$ service request slices, with the number of slices being greater than 0. If the value $\hat{s}_i^k = 0$, it signifies that the user is not utilizing the service. On the other hand, the value $\hat{s}_i^k = 1$ indicates that the service request is unsliced and will be processed solely by the Fog Node (FN).

Each SR is also associated with a size, which represents the magnitude of the SR or the size of the input data. Let $u_i(\hat{s}_k)$ denote the size of the SR of type s_k from user u_i. The size of the SR slices can vary, meaning that $u_i(\hat{s}_k, j_1)$ may not equal $u_i(\hat{s}_k, j_2)$ if j_1 and j_2 are different, where $0 < j1, j2 < \hat{s}_i^k$, and $\forall u_i \in U$. Additionally, the sizes of SRs differ, and the size of a specific SR slice is distinct from that of another SR slice. As a result, $u_1(\hat{s}_k, j)$ is not equal to $u_2(\hat{s}_k, j)$, where $\forall u_1, u_2 \in U$, and $i_1 \neq i_2$.

The demand for service s_k from all connected users at the Smart Gateway is represented by $SD(s_k)$. This value is calculated using the total number of users availing the service and the cumulative size of all SRs from all users. Mathematically, the service

demand (SD) takes into account the boolean variable $u_i(s_k)$, which indicates if a user is availing the service s_k, and $u_i(\hat{s}_k)$, which represents the size of the service requests of type s_k from user u_i.

$$SD(s_k) = \sum_{u_i \in U} [u_i(\hat{s}_k) \cdot u_i(s_k)] \tag{6}$$

By considering these factors, the Smart Gateway can better understand each user's specific needs, allowing for more efficient resource allocation and improved overall system performance.

3.3.1 Service Resource Usage

Service Resource Usage is an important aspect that illustrates the consumption of resources within fog and cloud environments. In these environments, resource usage for service requests (SR) is represented by $u_i(R_k^f(x))$ and $u_i(R_k^c(x))$, corresponding to the fog and cloud environments, respectively. Resource usage can be categorized into four distinct types: x = CPU, MEM, BANDWIDTH, ENERGY. These categories help in evaluating the performance and efficiency of the system across different resource dimensions. CPU usage is an indicator of the processing power being utilized, while MEM, or memory usage, demonstrates the consumption of the available memory capacity. BANDWIDTH, on the other hand, represents the network bandwidth being consumed during data transmission between the environments. Lastly, ENERGY usage signifies the energy consumption associated with running the infrastructure and processing service requests.

Additionally, when dividing the service request into slices, it is necessary to determine the resource usage for each individual slice. The resource usage for a specific slice can be calculated by dividing the size of that slice by the total size of the service request and then multiplying the result by the total resource usage for the entire service request. Mathematically,

$$u_i(R_{k,j}^f(x)) = u_i(R_k^f(x)) \cdot \frac{u_i(\hat{s}_{k,j})}{u_i(\hat{s}_k)} \tag{7}$$

$$u_i(R_{k,j}^c(x)) = u_i(R_k^c(x)) \cdot \frac{u_i(\hat{s}_{k,j})}{u_i(\hat{s}_k)} \tag{8}$$

3.3.2 Service Sensitivity Priority

The sensitivity level of a particular service s_k, is represented by P_k^s, with a value ranging between 0 and 1. The Smart Gateway is responsible for determining the sensitivity of each service. A higher P_k^s value indicates a more sensitive service. For instance, when

evaluating services, the processing of sensitive medical data has a higher sensitivity value compared to a service that manages user-created playlists of songs.

In this context, sensitivity may relate to factors such as security, privacy, and the critical nature of the service. Services with a higher sensitivity level may require additional precautions, resource allocation, or prioritization to ensure that they are handled properly and securely within the fog and cloud computing environments. By taking into account the sensitivity of each service, the Smart Gateway can effectively allocate resources and manage the services in a way that addresses the varying requirements and expectations of users, ultimately enhancing overall system performance and user satisfaction.

3.3.3 Service Priority

Service priority pertains to the significance assigned to a service when distributing resources among the service slices. This concept is distinct from service sensitivity priority, as it encompasses factors such as the distance and latency of connected users, as well as the relative service demand. The priority level for a service of type s_k, represented by P_k, can be determined using the following approach:

$$P_k = 0.5 \cdot \sum_{u_i \in U} [P_i \cdot u_i(\hat{s}_k)] + 0.5 \cdot \frac{SD(s_k)}{\sum_{s_j \in S} SD(s_j)} \tag{9}$$

The initial component of the calculation involves summing the priority of all users availing the service s_k, along with the size of the service requests. This user priority takes into account both the user's distance from the G and their communication latency. By considering these factors, the system can better address each user's specific needs and expectations. The second component of the service priority calculation represents the relative service demand. This is determined by comparing the demand of the current service request to the total service demand received by the G. By incorporating the relative service demand into the service priority calculation, the system can better allocate resources and manage services in a manner that is responsive to varying levels of demand, ultimately enhancing overall system performance and user satisfaction. Lastly, multiplying by 0.5 both sides of equation means that importance factors of both sides are equal.

3.4 DRL-based Service Placement

We discussed the challenge of effectively distributing services across fog and cloud computing environments in the previous section. Moving forward, this part will concentrate primarily on the suggested strategy, which involves adapting the DRL technique to include more practical parameters that cater to real-world situations. Numerous strategies, including heuristic methods, RL techniques, and DRL approaches, can be employed to achieve optimal service distribution in fog and cloud computing settings.

The heuristic approach involves using problem-solving methods based on experience, intuition, and trial-and-error to find solutions or make decisions in complex situations. Heuristics can help in generating quick, approximate solutions in scenarios where obtaining an optimal solution is computationally expensive or time-consuming. In the context of service distribution across fog and cloud computing environments, heuristic methods can be used to allocate resources, manage workload, and balance network traffic. These approaches often rely on predefined rules, historical data, or expert knowledge to guide the decision-making process. However, one of the major drawbacks of heuristic methods is their inability to adapt to dynamic changes in the environment, which is common in fog and cloud computing scenarios. Heuristic solutions may not always be optimal, as they can sometimes lead to suboptimal allocations or underutilization of resources. Additionally, heuristic approaches might struggle with scalability, as the complexity of the problem increases with the growth of the infrastructure. Heuristic algorithms can also suffer from a lack of flexibility, as updating or modifying their underlying rules can be challenging and time-consuming. In contrast to machine learning-based approaches like RL and DRL, heuristic methods may not be able to learn and improve their performance over time, limiting their long-term effectiveness. While heuristic approaches can provide some benefits for service distribution in fog and cloud computing environments, their limitations, such as adaptability, scalability, and flexibility, can lead to suboptimal performance compared to more advanced techniques like RL and DRL.

Reinforcement Learning (RL) is a subfield of machine learning that focuses on training agents to make decisions by interacting with an environment, with the aim of maximizing a cumulative reward signal. An RL agent learns through a trial-and-error process by continuously exploring the environment, taking actions, and receiving feedback in the form of rewards or penalties. The central idea behind RL is to find an optimal policy, which is a mapping from states to actions that maximizes the expected cumulative reward over time. In RL, the agent's learning process is typically modeled as a Markov Decision Process (MDP), which comprises a set of states, actions, transition probabilities, and reward functions. The Qfunction also known as the state-action value function, is a key concept in RL. It estimates the expected cumulative reward an agent will receive after taking a specific action in a given state and following an optimal policy thereafter. RL methods can adapt to dynamic environments and learn from experiences, making them suitable for various applications, including distributing services across fog and cloud computing environments. Q-learning, a popular RL algorithm, works by iteratively updating the Q-function values based on the agent's experiences, eventually converging to the optimal Q-function. The Q-function can be represented as a table (in tabular Q-learning). However, one of the primary challenges of RL methods is, as the state space and action space expand, the maintenance of the large tabular data becomes a significant limitation. One of the main issues with RL methods is that managing large amounts of tabular data becomes harder as the state and action spaces grow. To overcome

this, we can combine RL with other machine learning methods, like deep learning, to make better and more powerful algorithms such as DRL.

Instead of using a large matrix to store state-action pairs like Q-learning, deep learning is used to identify features through Artificial Neural Networks. This approach is used in many areas like image and video analysis and medical image prognosis. It's also applied in reinforcement learning by combining it with deep learning. This method helps the agent predict the best action to take by using a deep learning model with the current state as input rather than a specific state-action pair. Unlike Q-learning, which uses a Q-function to decide the agent's action, deep learning with reinforcement learning predicts an action based on all possible action prediction values. Figure 4 illustrates a comparison between the RL and DRL methods.

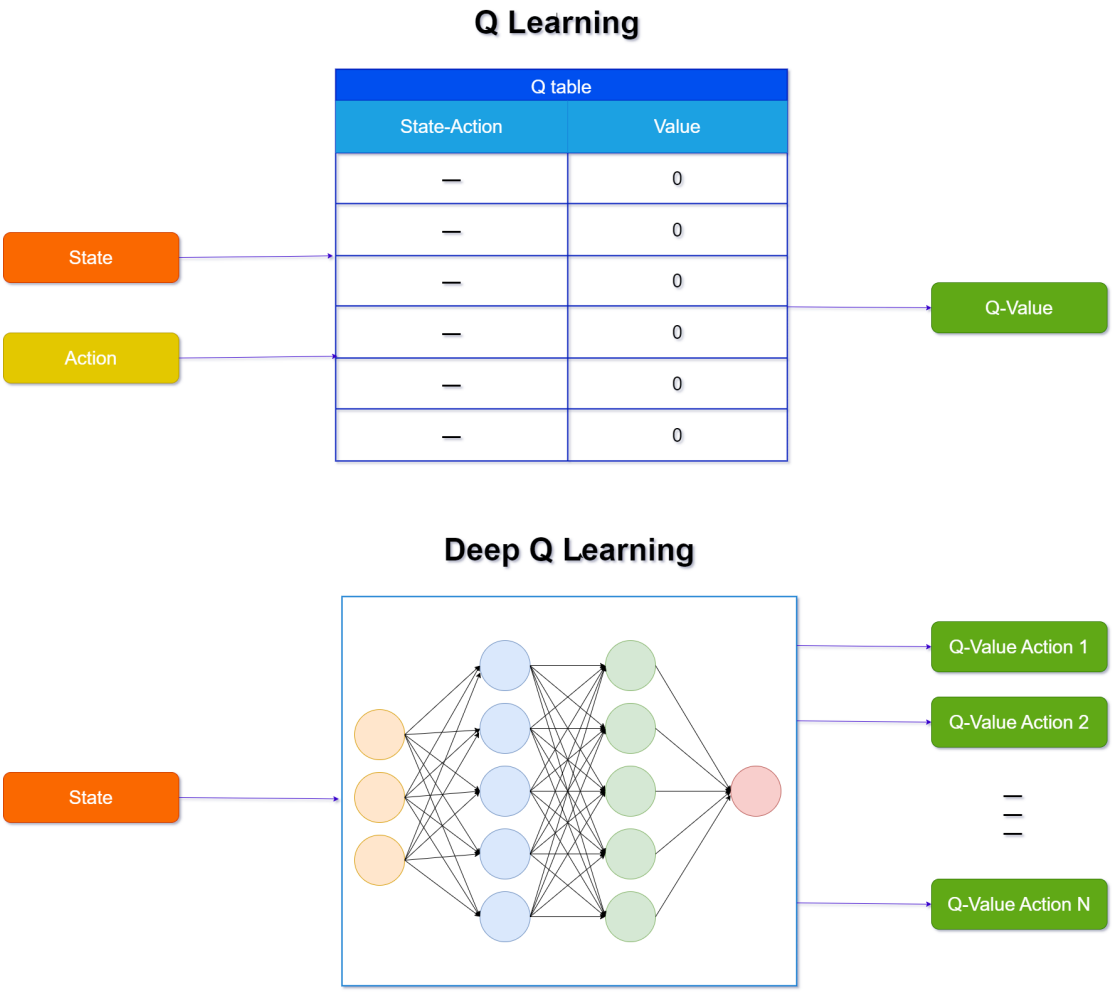

Figure 4. Comparison of RL and Deep-RL methods

3.4.1 State Representation

Defining the state space, as illustrated in Figure 4, is crucial. The state space encompasses all possible combinations of the states of all SRs. Let $\hat{S} = \hat{s}_1, \hat{s}_2, \hat{s}_3, ...,$ be the state space. A state of SR's type s_k signifies the environment in which it operates. Mathematically,

$$\hat{s}_i^k =< f(\hat{s}_i^k), c(\hat{s}_i^k) > \tag{10}$$

$$f(\hat{s}_i^k) + c(\hat{s}_i^k) = 1 \tag{11}$$

Here, $\hat{s}_i^k$ denotes the state of a SR s_k from user u_i. $f(\hat{s}_i^k)$ and $c(\hat{s}_i^k)$ are two boolean variables, indicating whether the SR is allocated to the fog or cloud environment, respectively. The value 1 represents the assignment of the SR to a specific environment, while 0 represents the opposite. Equation 11 must be satisfied for a valid service request assignment. For example, $\hat{s}_3^1 =< 1, 0 >$ implies that SR s_1 from u_3 is sent to the fog computing environment, which is this SR's state. Let $\hat{s}_{i,j}^k$ be the state of slice j of SR s_k from user u_i. Each SR slice's state indicates its assignment. Mathematically,

$$\hat{s}_{i,j}^k = \begin{cases} < 0, 0 > & \text{- the slice is not yet assigned to the neither fog nor cloud.} \\ < 1, 0 > & \text{- the slice is assigned to be processed in the fog environment.} \\ < 0, 1 > & \text{- the slice is assigned to be processed in the cloud environment.} \\ < 1, 1 > & \text{- invalid state} \end{cases} \tag{12}$$

By combining the states of all slices, the state of a SR $\hat{s}_{i,j}^k$ can be determined as follows:

$$\hat{s}_{i,j}^k =< \hat{s}_{i,1}^k, \hat{s}_{i,2}^k, ..., \hat{s}_{i,\hat{s}}^k > \tag{13}$$

Additionally, the entire SR's state can be expressed as:

$$\hat{s}_i^k =< \hat{s}_1^k, \hat{s}_2^k, ..., \hat{s}_n^k > \tag{14}$$

3.4.2 Action Selection

The action space encompasses all potential actions the agent can take, given its present state. Denoted by $\hat{A}$, the action space characterizes the range of possible decisions available to the agent. The current action within the action space is represented as $\hat{a}_t$, and its definition is provided in Equation, below:

$$\hat{a}_t =< \hat{a}_f, \hat{a}_c > \tag{15}$$

In this equation, $\hat{a}_f$ signifies actions where the function is allocated to the fog node, while $\hat{a}_c$ represents actions involving the assignment of the function to the cloud computing environment. By considering all possible actions within the action space, the agent can make informed decisions that optimize its objectives while navigating the fog and cloud computing environments.

3.4.3 Reward Function

3.5 Summary

4 Experiment

4.1 Experimental Setup

4.2 Experiment Applications

The original versions of applications were modified to satisfy the problem formulation, and the revised editions used in this sction are available in the project repository. This section explains the services available in the fog and cloud environment.

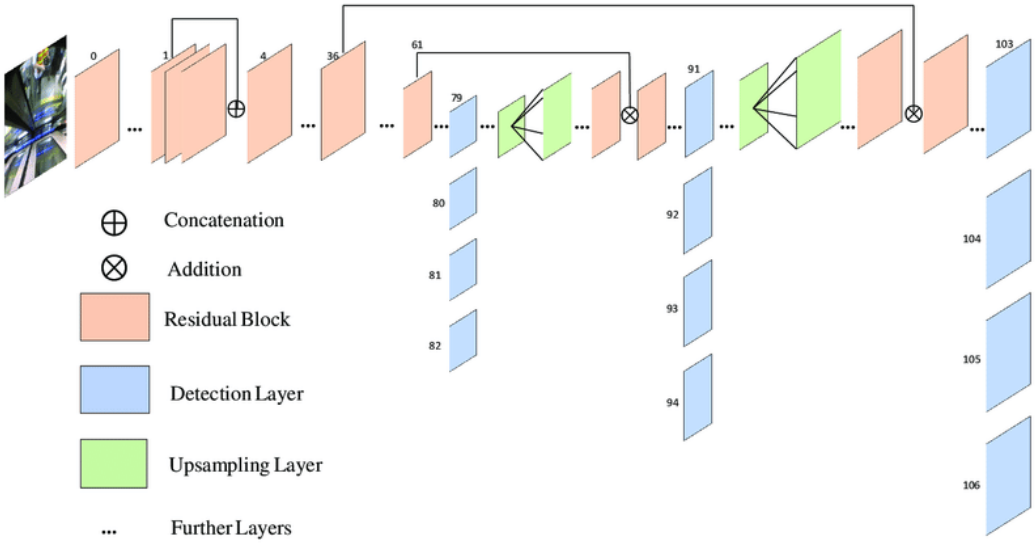

Figure 5. Network architecture of YOLOv3 [DLLL20].

4.2.1 Application 1: Deep learning based object classification

YOLOv3 (You Only Look Once) is real-time object detection algorithm that uses a single convolutional neural network (CNN) to detect objects in an image. It works by dividing an input image into a grid of cells, and predicting bounding boxes and class probabilities for each cell. Each bounding box consists of four coordinates (x, y, width, height) and a confidence score, which represents the likelihood that the box contains an object. Class probabilities are predicted for each box, indicating the likelihood that the object in the box belongs to a certain class. To improve the accuracy of object detection, YOLOv3 uses a few key techniques. One is feature pyramid networks, which use a pyramid of features extracted from multiple layers of the network to improve the detection of objects at different scales as shown in Figure 5. Another technique is residual connections, which

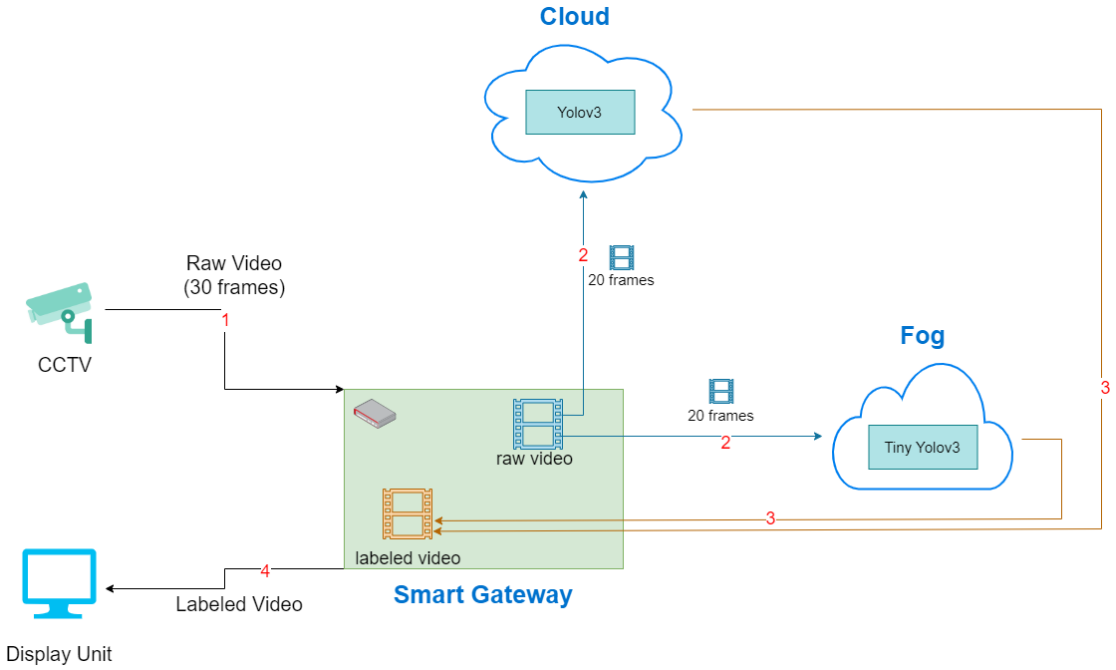

Figure 6. Architecture of Application 1: Deep learning based object classification

allow gradients to flow more easily through the network during training, leading to better accuracy.

This application is a suitable choice for Fog environment, it can be deployed to detect objects in order to reduce communication delays in fog computing. The Algorithm 1 represents how Yolov3 used in this application. Additionally, The Figure 6 illustrates an architecture of deep learning based object classification application.

Algorithm 1: Deep learning based object classification

Input: Video to be analyzed
Result: Detected objects in the video

```
Load the pre-trained YOLOv3 object detection model;
foreach frame in video do
    Extract image from frame;
    Perform object detection using YOLOv3;
    Draw bounding boxes around detected objects;
    Add labels to bounding boxes;
    Insert the annotated image back into the frame;
    Add the frame to the output video;
```

4.2.2 Application 2: Text-audio synchronisation or forced alignment

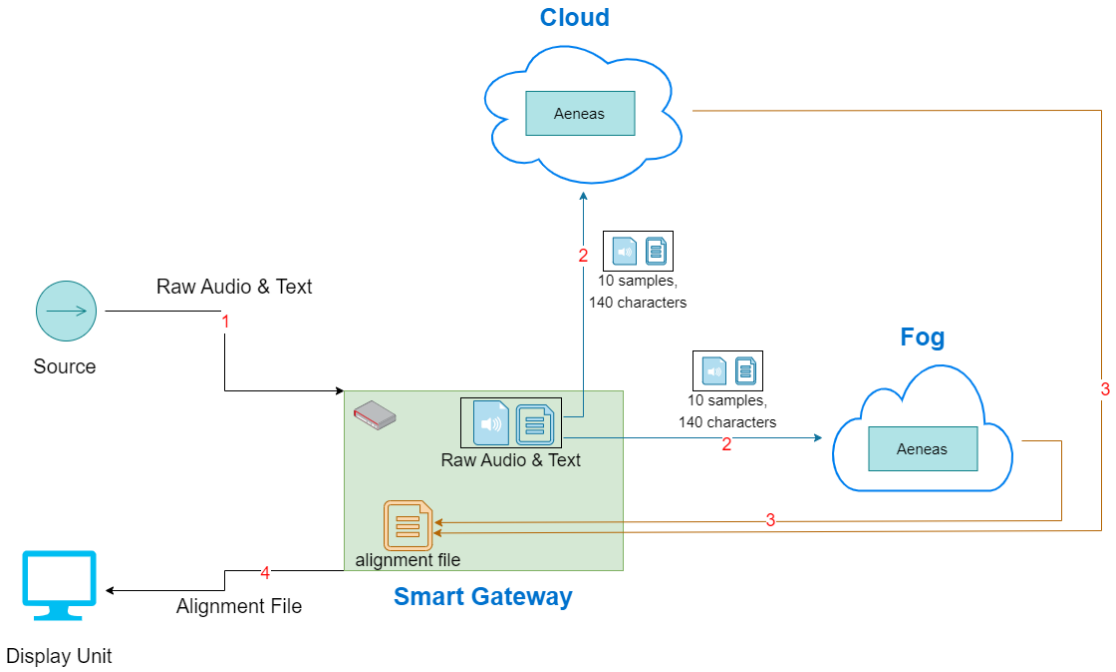

Figure 7. Architecture of Application 2: Text-audio synchronisation or forced alignment

Aeneas is a text-alignment library that aligns text with audio or video content. As shown in Algorithm 2, the library create sync map, allowing users to jump to the correspanding point in the audio or video when clicking on a specific word or phrase in the text. Aeneas works by first converting the audio or video into a time-aligned phonetic transcript, known as an "alignment graph." It then generates a time-aligned version of the text, known as a "fragmented transcript," based on the words and timing information in the alignment graph. Next, Aeneas performs a process called "forced alignment," which uses statistical models to align the transcript to the audio or video. This process allows for a precise time-based synchronization of the text and media content. The resulting output of Aeneas can be an XML, JSON, SRT files that includes a detailed mapping of the text and media content, including time codes and other metadata. The Figure 7 illustrates an architecture of text-audio synchronisation or forced alignment application.

4.2.3 Application 3: Speech-to-text conversion

PocketSphinx is a lightweight speech recognition engine. It is based on the Sphinx-4 open source speech recognition system, which uses Hidden Markov Models (HMMs) to model the acoustic features of speech. PocketSphinx is capable of recognizing speech

Algorithm 2: Text-audio synchronisation or forced alignment

Input: Text and Audio Files
Output: Alignment File (*.srt)

```
1 Procedure align(TextFile, AudioFile)
      /* Initialize Aeneas object                                  */
2     aeneas = Aeneas(language="en")
      /* Create the sync map                                       */
3     syncmap = SyncMap()
      /* Perform the alignment                                     */
4     aeneas.execute(text=TextFile, audio=AudioFile, sync=syncmap)
      /* Save the sync map as an alignment file                    */
5     syncmap.writetofile("alignment.srt")
6     Return the saved alignment file as an output;
```

Algorithm 3: Speech-to-text conversion

Input: Audio file to be transcribed
Result: Transcription of the audio file

1 Load the audio file;
2 Initialize the PocketSphinx recognizer;
3 Set the language model and dictionary;
4 Process the audio file in chunks;
5 Feed the chunks to the recognizer;
6 Get the transcription from the recognizer;
7 Output the transcription;

in a variety of languages, and can be used for tasks such as voice search, dictation, and command recognition. The PocketSphinx library provides a set of Python APIs for developers to use in their applications. As shown in Algorithm 3, to use PocketSphinx, firstly start by configuring a decoder, which specifies the acoustic and language models to be used for speech recognition. The audio input is then passed to the decoder, which performs a series of signal processing and feature extraction operations to produce a stream of speech recognition hypotheses. These hypotheses are then scored and ranked according to their likelihood of being correct, and the most likely hypothesis is returned as the recognized text. The Figure 8 illustrates an architecture of speech-to-text conversion application.

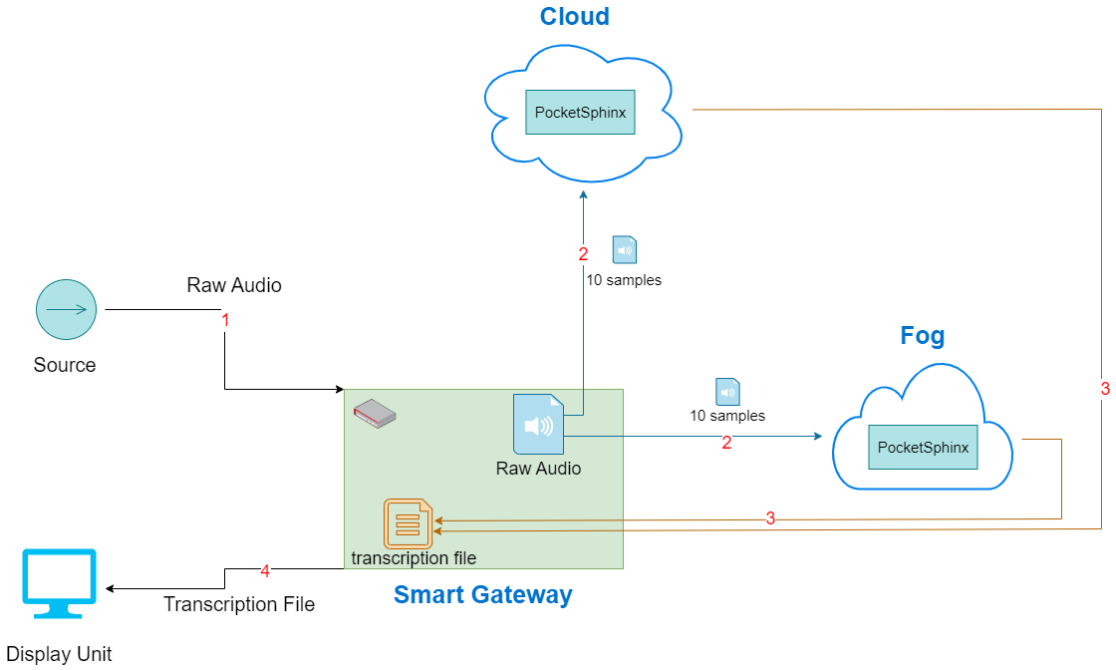

Figure 8. Architecture of Application 3: Speech-to-text conversion

4.3 Summary

5 Results

5.1 Summary

6 Conclusion

References

[AH14] Mohammad Aazam and Eui-Nam Huh. Fog computing and smart gateway based communication for cloud of things. In *2014 International conference on future internet of things and cloud*, pages 464–470. IEEE, 2014.

[AHU+19] Md Golam Rabiul Alam, Mohammad Mehedi Hassan, Md ZIa Uddin, Ahmad Almogren, and Giancarlo Fortino. Autonomic computation offloading in mobile edge for iot applications. *Future Generation Computer Systems*, 90:149–157, 2019.

[APH+16] Aymen Abdullah Alsaffar, Hung Phuoc Pham, Choong-Seon Hong, Eui-Nam Huh, and Mohammad Aazam. An architecture of iot service delegation and resource allocation based on collaboration between fog and cloud computing. *Mobile Information Systems*, 2016, 2016.

[BF17] Antonio Brogi and Stefano Forti. Qos-aware deployment of iot applications through the fog. *IEEE internet of Things Journal*, 4(5):1185–1192, 2017.

[BLK22] Hayat Bashir, Seonah Lee, and Kyong Hoon Kim. Resource allocation through logistic regression and multicriteria decision making method in iot fog computing. *Transactions on Emerging Telecommunications Technologies*, 33(2):e3824, 2022.

[BMNZ14] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. *Big data and internet of things: A roadmap for smart environments*, pages 169–186, 2014.

[DGC+16] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya. Fog computing: Principles, architectures, and applications. In *Internet of things*, pages 61–75. Elsevier, 2016.

[DLL+16] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181, 2016.

[DLLL15] Ruilong Deng, Rongxing Lu, Chengzhe Lai, and Tom H. Luan. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *2015 IEEE International Conference on Communications (ICC)*, pages 3909–3914, 2015.

[DLLL20] Yuan Dai, Weiming Liu, Haiyu Li, and Lan Liu. Efficient foreign object detection between psds and metro doors via deep neural networks. *IEEE Access*, 8:46723–46734, 2020.

[DS19] Chinmaya Kumar Dehury and Satish Narayana Srirama. Personalized service delivery using reinforcement learning in fog and cloud environment. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*, pages 522–529, 2019.

[DS20] Chinmaya Kumar Dehury and Satish Narayana Srirama. An efficient service dispersal mechanism for fog and cloud computing using deep reinforcement learning, 2020.

[DSMP19] Tanissia Djemai, Patricia Stolf, Thierry Monteil, and Jean-Marc Pierson. A discrete particle swarm optimization approach for energy-efficient iot services placement over fog infrastructures. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 32–40. IEEE, 2019.

[Ela19] Hanan Elazhary. Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. *Journal of Network and Computer Applications*, 128:105–140, 2019.

[FPMJMV18] Francisco-Javier Ferrández-Pastor, Higinio Mora, Antonio Jimeno-Morenilla, and Bruno Volckaert. Deployment of iot edge and fog computing technologies to develop smart building services. *Sustainability*, 10(11):3832, 2018.

[FSM20] Peter Farhat, Hani Sami, and Azzam Mourad. Reinforcement r-learning model for time scheduling of on-demand fog placement. *the Journal of Supercomputing*, 76:388–410, 2020.

[FSV+19] Xiang Fei, Nazaraf Shah, Nandor Verba, Kuo-Ming Chao, Victor Sanchez-Anguix, Jacek Lewandowski, Anne James, and Zahid Usman. Cps data streams analytics based on machine learning for cloud and fog computing: A survey. *Future generation computer systems*, 90:435–450, 2019.

[GRN20] Pegah Gazori, Dadmehr Rahbari, and Mohsen Nickray. Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. *Future Generation Computer Systems*, 110:1098–1115, 2020.

[HKTG20] Pejman Hosseinioun, Maryam Kheirabadi, Seyed Reza Kamel Tabbakh, and Reza Ghaemi. A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *Journal of Parallel and Distributed Computing*, 143:88–96, 2020.

[HWC+17] Jianhua He, Jian Wei, Kai Chen, Zuoyin Tang, Yi Zhou, and Yan Zhang. Multitier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal*, 5(2):677–686, 2017.

[JSPP21] Saeed Javanmardi, Mohammad Shojafar, Valerio Persico, and Antonio Pescapè. Fpfts: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for internet of things devices. *Software: practice and experience*, 51(12):2519–2539, 2021.

[Kim16] HS Kim. Fog computing and the internet of things: extend the cloud to where the things are. *Int. J. Cisco*, 2016.

[LGL+15] Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

[LHD+20] Haodong Lu, Xiaoming He, Miao Du, Xiukai Ruan, Yanfei Sun, and Kun Wang. Edge qoe: Computation offloading with deep reinforcement learning for internet of things. *IEEE Internet of Things Journal*, 7(10):9255–9265, 2020.

[MSRB19] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. Quality of experience (qoe)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing*, 132:190–203, 2019.

[MWT+19] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal De Lara, and Blesson Varghese. Defog: fog computing benchmarks. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 47–58, 2019.

[OPR18] Alexandru Iulian Orhean, Florin Pop, and Ioan Raicu. New scheduling approach using reinforcement learning for heterogeneous distributed systems. *Journal of Parallel and Distributed Computing*, 117:292–302, 2018.

[PJ19] HMB Priyabhashana and KPN Jayasena. Data analytics with deep neural networks in fog computing using tensorflow and google cloud platform.

In *2019 14th Conference on Industrial and Information Systems (ICIIS)*, pages 34–39. IEEE, 2019.

[RN20] Dadmehr Rahbari and Mohsen Nickray. Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications*, 13:104–122, 2020.

[RSI+19] Hina Rafique, Munam Ali Shah, Saif Ul Islam, Tahir Maqsood, Suleman Khan, and Carsten Maple. A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing. *IEEE Access*, 7:115760–115773, 2019.

[RZA21] Xiaojun Ren, Zhijun Zhang, and Seyedeh Maryam Arefzadeh. An energy-aware approach for resource managing in the fog-based internet of things using a hybrid algorithm. *International Journal of Communication Systems*, 34(1):e4652, 2021.

[SCAF+19] Mennan Selimi, Llorenç Cerdà-Alabern, Felix Freitag, Luís Veiga, Arjuna Sathiaseelan, and Jon Crowcroft. A lightweight service placement approach for community network micro-clouds. *Journal of Grid Computing*, 17:169–189, 2019.

[SDL20] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)*, 53(3):1–35, 2020.

[SDV18] Prasan Kumar Sahoo, Chinmaya Kumar Dehury, and Bharadwaj Veeravalli. Lvrm: On the design of efficient link based virtual resource management algorithm for cloud platforms. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):887–900, 2018.

[SMBMT+18] V.B. Souza, X. Masip-Bruin, E. Marín-Tordera, S. Sànchez-López, J. Garcia, G.J. Ren, A. Jukan, and A. Juan Ferrer. Towards a proper service placement in combined fog-to-cloud (f2c) architectures. *Future Generation Computer Systems*, 87:1–15, 2018.

[SMW18] Hamed Shah-Mansouri and Vincent W. S. Wong. Hierarchical fog-cloud computing for iot systems: A computation offloading game. *IEEE Internet of Things Journal*, 5(4):3246–3257, 2018.

[SNSD17] Olena Skarlat, Matteo Nardelli, Stefan Schulte, and Schahram Dustdar. Towards qos-aware fog service placement. In *2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC)*, pages 89–96. IEEE, 2017.

[SWHL16] Ivan Stojmenovic, Sheng Wen, Xinyi Huang, and Hao Luan. An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience*, 28(10):2991–3005, 2016.

[TD17] Mohit Taneja and Alan Davy. Resource aware placement of iot application modules in fog-cloud computing paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1222–1228. IEEE, 2017.

[TZZ+18] 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, 2018.

[WL19] Juan Wang and Di Li. Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. *Sensors*, 19(5):1023, 2019.

[WLLC19] Yimeng Wang, Yongbo Li, Tian Lan, and Nakjung Choi. A reinforcement learning approach for online service tree placement in edge computing. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2019.

[XHZS19] Jiuyun Xu, Zhuangyuan Hao, Ruru Zhang, and Xiaoting Sun. A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. *IEEE Access*, 7:116218–116226, 2019.

[YHQL15] 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. IEEE, 2015.

[YJR+18] Anila Yasmeen, Nadeem Javaid, Obaid Ur Rehman, Hina Iftikhar, Muhammad Faizan Malik, and Fatima J Muhammad. Efficient resource provisioning for smart buildings utilizing fog and cloud based environment. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 811–816. IEEE, 2018.

[YMY20] Seema Yadav, Rakesh Mohan, and Pradeep Kumar Yadav. Task allocation model for optimal system cost using fuzzy c-means clustering technique in distributed system. *Ingénierie des Systèmes d Inf.*, 25(1):59–68, 2020.

[YNG19] Vinita Yadav, BV Natesha, and Ram Mohana Reddy Guddeti. Gapso: Service allocation in fog computing environment using hybrid bio-inspired algorithm. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 1280–1285. IEEE, 2019.

[YQL15] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *Wireless Algorithms, Systems, and Applications: 10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings 10*, pages 685–695. Springer, 2015.

[YXZ18] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. Application provisioning in fog computing-enabled internet-of-things: A network perspective. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 783–791, 2018.

[ZJJ+18] Saman Zahoor, Sakeena Javaid, Nadeem Javaid, Mahmood Ashraf, Farruh Ishmanov, and Muhammad Khalil Afzal. Cloud–fog–based smart grid model for efficient resource management. *Sustainability*, 10(6):2079, 2018.

[ZLY+18] Qingchen Zhang, Man Lin, Laurence T Yang, Zhikui Chen, Samee U Khan, and Peng Li. A double deep q-learning model for energy-efficient edge scheduling. *IEEE Transactions on Services Computing*, 12(5):739–749, 2018.

Appendix

I. List of Figures

II. List of Tables

III. Source Code

The source code of applications and developed algorithms for this study is located in repository https://github.com/chinmaya-dehury/AI4FogCloudServiceDisperse.The access to the repository could be granted by sending an email to chinmaya.dehury@ut.ee

IV. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Jeyhun Abbasov**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   AI-powered cloud usage controlling system in smart home environment,

   supervised by Prof. Dr. Chinmaya Dehury.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Jeyhun Abbasov
dd/mm/yyyy