PERFORMANCE ANALYSIS AND ENHANCEMENT OF DELAY-SENSITIVE AND ENERGY-HUNGRY MOBILE AI APPLICATIONS WITH EDGE COMPUTING

by

Anik Mallik

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2024

Approved by:

____

Dr. Jiang Xie

____

Dr. Ahmed Arafa

____

Dr. Weichao Wang

____

Dr. Pu Wang

ABSTRACT

ANIK MALLIK. Performance Analysis and Enhancement of Delay-sensitive and Energy-hungry Mobile AI Applications with Edge Computing. (Under the direction of DR. JIANG XIE)

Artificial intelligence (AI) has enabled a new paradigm of smart applications, such as extended reality (XR), which comprises mobile augmented reality (MAR), mixed (MR), and virtual reality (VR), which have very stringent latency requirements, especially for applications on mobile devices (e.g., smartphones, wearable devices, and autonomous vehicles). Edge computing-assisted mobile AI systems have emerged as effective ways to support computation-intensive and latency-sensitive applications for mobile devices due to the offloading capability of heavy computational burdens. However, the high mobility of users and instability in wireless networks decrease the overall Quality-of-Service (QoS) of an edge-AI application running on mobile devices with non-linear battery discharge properties.

This dissertation presents a comprehensive experimental study of mobile AI applications, considering different DNN models and processing sources, focusing on computational resource utilization, delay, and energy consumption. Additionally, a novel Gaussian process regression-based general predictive energy model is proposed based on DNN structures, computation resources, and processors, which can predict the energy for each complete application cycle irrespective of device configurations.

In addition, a novel performance analysis modeling framework of XR applications is proposed, considering heterogeneous wireless networks and using experimental data collected from testbeds designed specifically for this research. A comprehensive performance analysis model is challenging to design due to the dependence of the performance metrics on several difficult-to-model parameters, such as computing resources and hardware utilization of XR and edge devices, which are controlled by their operating systems, and the heterogeneity in devices and wireless access networks. These challenges and ways to overcome them are also presented in detail.

Following the performance analysis model, performance enhancement of mobile AI applications is also proposed in this research. This dissertation focuses on the high mobility of users in connected and autonomous vehicles (CAVs). Time-sensitive information update services are necessary for these CAV-AI applications to ensure the safety of people and assets, and satisfactory entertainment applications. However, information from roadside sensors and nearby vehicles can get delayed in transmission due to the high mobility of vehicles. This research proposes a novel periodic predictive AoI-based service aggregation method for CAVs, which can process the information updates according to their update cycles by maintaining a satisfactory latency and data sequencing success rate (DSSR) for CAV-AI applications.

Furthermore, an unstable wireless network poses a critical challenge for real-time mobile AI applications. An H.264 video encoding-based edge-MAR system is proposed, with a focus on network conditions, resource utilization, detection accuracy, and energy consumption of various mobile devices. This extensive study provides essential guidelines for network- and energy-aware H.264 video encoding-based Edge-MAR system design to overcome the challenges posed by unstable wireless networks.

Finally, a novel deep reinforcement learning-based smart edge-MAR system – Reinforced Edge-Assisted Learning (REAL), is proposed in this research, where the edge server is exploited to unleash its potential by providing smart and dynamic MAR processing decisions to the devices based on dynamically changing system states, such as wireless link qualities and battery energy levels of mobile devices. The novelty of REAL lies in solving the complex state-transition problem in a stochastic environment through online soft actor-critic learning and delivering reward-based actions to mobile devices to improve the end-to-end latency, energy consumption, accuracy, and offloaded data size collectively.

DEDICATION

To my son, Aditya.

You will take a look at your father's work someday,

And realize that sometimes it takes generations to fulfill a dream.

Live your life to the fullest, my child.

ACKNOWLEDGEMENTS

I would like to extend my sincerest and deepest appreciation to my Ph.D. advisor and mentor, **Professor Jiang Xie**, for her unwavering guidance and supervision throughout this period. Her scientific methods and strong work ethic have shaped my personal growth and research abilities. A doctoral journey is challenging for every student, but Professor Xie has made this journey for me smooth and full of learning. Without any hesitation, I can say, she is the most incredible mentor I have ever had.

Additionally, I am grateful to my committee members, **Dr. Ahmed Arafa, Dr. Weichao Wang, and Dr. Pu Wang**, for generously dedicating their time and providing valuable advice. I also appreciate the financial support from the Graduate Assistant Support Plan (**GASP**) grant at the University of North Carolina at Charlotte, the research assistantships provided by the National Science Foundation (**NSF**), and funds from **Toyota Motor North America**, making this work possible.

On a different note, I believe the idea of a "self-made man" is a myth. If I consider finishing this dissertation an achievement, I owe thanks to all the people surrounding me. My parents and friends have created a support system without which I could not have come this far. My lab mates, **Dr. Moinul Hossain, Dr. Haoxin Wang, Munmun Talukder, and Md. Toufiqur Rahman** constantly pushed me to go beyond my comfort zone and strive for excellence in research. Being an international graduate student in the United States is challenging. My mentor, my family, and all my friends in my home country and the US have helped me navigate this journey.

At the heart of this support system are my beautiful wife, **Susmita Bhowmik**, and my loving son, **Aditya Mallik**, who have consistently inspired me to accomplish great things. Mere words cannot adequately express my gratitude for my wife's sacrifices. I thank my family for celebrating every success and standing by me through every failure I ever had.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artifical Intelligence.
AoI	Age-of-Information.
AP	Access Point.
AR	Augmented Reality.
CAV	Connected and Autonomous Vehicles.
CAV-AI	Connected and Autonomous Vehicle Artificial Intelligence applications.
CDMA	Code-Division Multiple Access.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
DNN	Deep Neural Network.
DoF	Degree-of-Freedom.
DRL	Deep Reinforcement Learning.
DSSR	Data Sequencing Success Rate.
Edge-AI	Edge Computing-Assisted Artificial Intelligence.
Edge-MAR	Edge Computing-Assisted Mobile Augmented Reality.
EPAM	Energy Prediction of AI Applications in Mobile Devices.
FPS	Frame Per Second.
FVHO-XR	Fast Vertical Handoff for Extended Reality.

Gen-AI	Generative Artificial Intelligence.
GPR	Gaussian Process Regression.
GPU	Graphics Processing Unit.
HMD	Head-Mounted Device.
HO	Handoff.
IoT	Internet-of-Things.
LLM	Large Language Model.
LSTM	Long Short-Term Memory.
MANET	Mobile Ad-Hoc Network.
MAR	Mobile Augmented Reality.
MEC	Mobile Edge Computing.
MR	Mixed Reality.
NLP	Natural Language Processing.
NNAPI	Neural Network Application Programming Interface.
NPU	Neural Processing Unit.
QoS	Quality-of-Service.
REAL	Reinforced Edge-Assisted Learning.
RL	Reinforcement Learning.
RoI	Relevance-of-Information.
RSSI	Received Signal Strength Indicator.

SAC	Soft-Actor-Critic.
SCAR	Speed-to-Coverage Area Ratio.
SoC	System-on-Chip.
UAV	Unmanned Aerial Vehicle.
UMTS	Universal Mobile Telecommunications System.
VANET	Vehicular Ad-Hoc Network.
VR	Virtual Reality.
WLAN	Wireless Local Area Network.
XR	Extended Reality.

CHAPTER 1: INTRODUCTION

1.1 Background on AI Applications in Mobile Devices using Edge Computing in a Heterogeneous Wireless Network

Artificial intelligence (AI) is shaping every aspect of human lives nowadays. Furthermore, mobile devices, i.e., smartphones, tablets, wearable devices, augmented reality (AR) glasses, mixed reality (MR) and virtual reality (VR) headsets, Internet-of-Things (IoT) devices, autonomous driving systems (ADSs) and unmanned aerial vehicles (UAVs), are heavily invested in AI applications, having cellular networks, edge, and cloud computing in the backbone. The number of devices using AI applications is increasing day by day, which is considered to be a multi-billion dollar at this instance. Fig. 1.1 shows some primary user devices equipped with mobile AI applications.

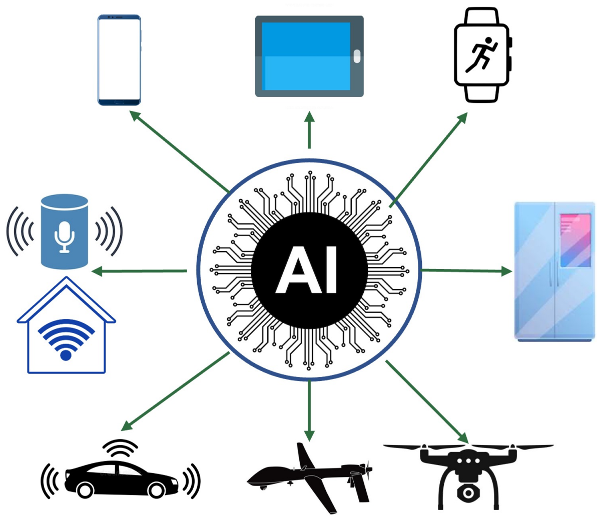


Figure 1.1: User devices in mobile AI.

AI applications consume considerably high energy and memory of these devices. How AI uses these resources defines a device's potential to interact with wireless networks. Therefore, it is crucial to understand the characteristics of AI applications running on a mobile device, which pushes back to the question — how can the performance of mobile AI applications be accurately analyzed irrespective of device configurations to ensure better service and user experience?

AI applications' latency and energy consumption may depend on various properties of a system. First, the AI models, crafted in specific ways to fit mobile devices due to the models' high computation and energy requirements, impact the applications' behaviors. Research works suggest accelerating the processing time of deep neural networks (DNNs) by quantizing [3], which is a compression technique run on DNN models that can reduce the model size by converting some tensor operations to integers from floating points or reducing the weights or parameters in a model, but at the cost of degraded accuracy. Quantized DNN (Q-DNN) models are generally investigated for vision-based applications, the most thriving areas of AI. Second, mobile AI is not limited to vision applications only. Modern-day mobile devices are rigged with non-vision applications as well, such as intelligent recommendations, natural language processing (NLP), smart reply, speech recognition, and speech-to-text conversion. While most of the research focuses on applications based on computer vision, acquiring a thorough knowledge of mobile AI is only possible by including non-vision applications. Third, the processing source used to run the AI models affects their performance. Besides central processing units (CPUs) with high processing speeds, some devices are now equipped with graphics processing units (GPUs), which enables DNN models to run faster than ever, especially for vision applications [4]. In addition, neural network application programming interfaces (NNAPI) are also developed to make the processing of DNN models faster, using CPUs, GPUs, or neural processing units (NPUs) [5]. These state-of-the-art technologies are researched for mobile AI only to improve inference latency. Lastly, the hardware configuration of mobile devices is distinctive and contributes to energy consumption with a unique signature.

The system-on-chip (SoC), CPU/GPU parameters, and memory dictate how an AI application runs on a specific device. Hence, an experimental study is required to analyze mobile AI performance with unique devices and AI configurations.

Vision-based applications of mobile AI generally refer to object detection, image classification, image segmentation, super resolution, image superposition, and so on. These applications are heavily used in extended reality (XR) applications. XR is comprised of three broad concepts: AR, MR, and VR [6]. AR is an interactive experience of a real-world environment where objects in the real world are enhanced by computer-generated perceptual information. When AR is used with a mobile device, the concept is called mobile AR (MAR). Additionally, VR is an immersive experience where the user can see a virtual world and objects around him while wearing a head-mounted device (HMD). MR is a combination of both AR and VR, where users are able to see the real world along with virtual objects. Fig. 1.2 shows the domains of mobile AI considered in this research. This dissertation presents research work on performance analysis and enhancement of mobile AI, XR applications, and edge-assisted MAR applications.

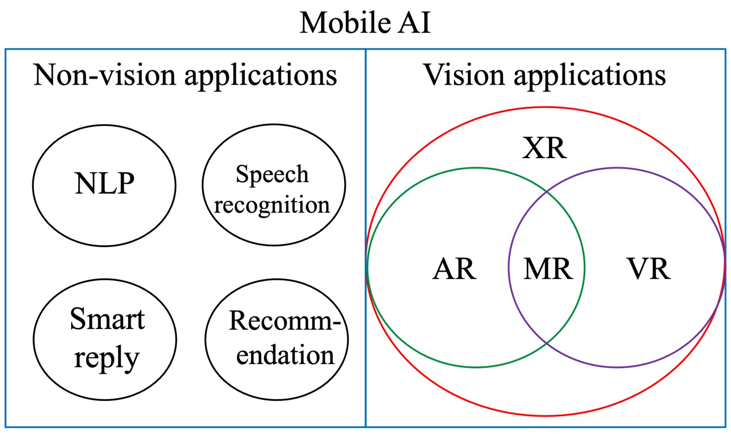


Figure 1.2: Domains of mobile AI considered in this research.

Since XR applications combine all the aspects of AR, MR, and VR, they have to follow the stringent Quality-of-Service (QoS) requirements imposed by each of these services, such as data rate, refresh rate, frame resolution, freshness of data, latency, and battery support. However, XR devices are usually lightweight and equipped with low computational capability to run high-complexity tasks, such as deep learning applications, which usually cause a higher latency and energy consumption [7]. To meet the QoS requirements, cloud and edge computing are often offered as solutions to reduce latency and energy consumption as well as increase service accuracy [8, 9]. Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data, where heavy computational tasks are offloaded to a nearby edge server from a mobile device. The server completes the computation and sends the result back to the mobile [10, 11]. In this way, mobile devices can save battery life. Mobile vision applications, including XR, can now be provided to mobile users being edge-assisted [12]. While edge computing can reduce the computational load on XR client devices, it is also imperative to maintain the expected QoS of XR applications, which brings attention to the performance analysis of XR applications. A unified performance analysis model for XR applications is necessary since XR is the next-generation medium for communication, entertainment, manufacturing, defense, automobiles, education, and healthcare [13]. However, there has been no significant research work on analyzing the comprehensive performance of XR applications considering user mobility in a heterogeneous wireless network and external control information from sensors and devices.

One prominent application area of XR is in connected and autonomous vehicles (CAVs). Recent advancements in CAV applications have enabled vehicles and roadside sensors in the vicinity to share and receive information updates on numerous tasks, such as dynamic map updates, probable collision detection, and obstacle recognition [14]. The communication between vehicles and sensors is used to run both safety (e.g., collision, asset damages, and public safety) and entertainment applications (e.g., cooperative extended reality). In these applications, broadcasting by roadside sensors and nearby connected vehicles is a common way to convey information to an ego vehicle (i.e., the vehicle in consideration or the target vehicle) [15]. The protocol design and security enhancement of such broadcast messages have been well-studied for vehicular ad-hoc networks (VANETs) and CAVs [16]. However, new challenges arise when more sensors and vehicles are connected to a single vehicle. For instance, a CAV has to determine whether and when it needs the information update service from another sensor or vehicle. Additionally, with mobility, the Age-of-Information (AoI) or the freshness of information received from different sources will vary as well, which may significantly impact the CAV application performance.

Furthermore, there is an increasing concern about the network resources, such as bandwidth, used to support edge-assisted MAR or edge-MAR applications. A straightforward way to minimize network resource utilization is to reduce the data size transmitted to and from the edge server over the wireless network. From experimental studies, it is observed that a 400×400 image frame sent to an edge server usually takes about 1.2 MB of data, which results in 36 MB per second on average if frames are being sent at a rate of 30 frames per second (fps). With the increase in the number of edge-MAR users, the large amount of data being transmitted may cause severe delays, which is in contrast to the requirements of real-time applications. Video frame encoding schemes like H.264 help compress the source data and reduce the data size [17]. H.264 encoding compresses the redundant background while keeping the objects of interest in consecutive frames intact. Therefore, there is an opportunity to explore this encoding scheme in MAR. However, with constantly varying wireless link quality, there comes another challenge with the non-linear discharge property of mobile devices' batteries.

The experiments done in this research show that the batteries of mobile devices do not discharge in a linear pattern. After a certain level of battery capacity is reached, the discharge curve becomes even steeper. This property requires a smart decision framework for offloading and data processing tasks in edge-assisted mobile XR or MAR services. The framework should be able to decide dynamically whether to offload or how to process the input data to reduce the network and energy effects to maximize the overall Quality-of-Service (QoS) of an XR application by reducing latency, energy consumption, and data size along with maintaining the benchmark service accuracy.

1.2 Problem Statement

1.2.1 Unique Latency and Energy Consumption Characteristic of Mobile AI

While mobile AI is often concluded as "no one-size-fits-all solution" [18], it is the responsibility of the research community to provide the developers with precise measurement data and a way to predict energy consumption. This research shows that the power varies for the same device with the change in processing sources (Fig. 1.3). The granularity of power consumption over a unit period of time needs to be measured to develop a predictive energy model, which is not provided by the current works. Battery profilers provided by third-party applications do not support precise energy data collection [7]. Hence, the use of an external power measurement tool becomes necessary [19].

Moreover, DNN models with different sizes and layers do not have a similar impact on the latency, energy, and memory usage, which is presented in Fig. 1.4, where it is evident that the correlation among latency, energy, and memory is not linear at all. An interesting observation here is that the Quantized EfficientNet model causes high latency and energy despite using the lowest memory, due to its compatibility issues with NNAPI, which is described in detail in Sec. 3.3. Therefore, data from physical testbeds are required to validate this relation before proposing a predictive energy model. Existing research works do not involve experimental measurements of all the parameters necessary to define the performance of DNN models on a mobile device, such as latency, energy consumption, confidence scores, memory usage based on the

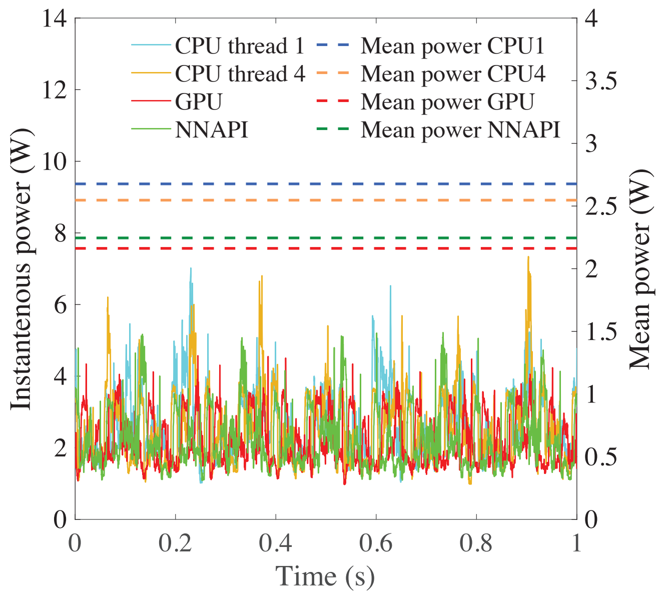


Figure 1.3: Power consumption by different processors for the same time interval for MobileNet Float on Huawei Mate40Pro.

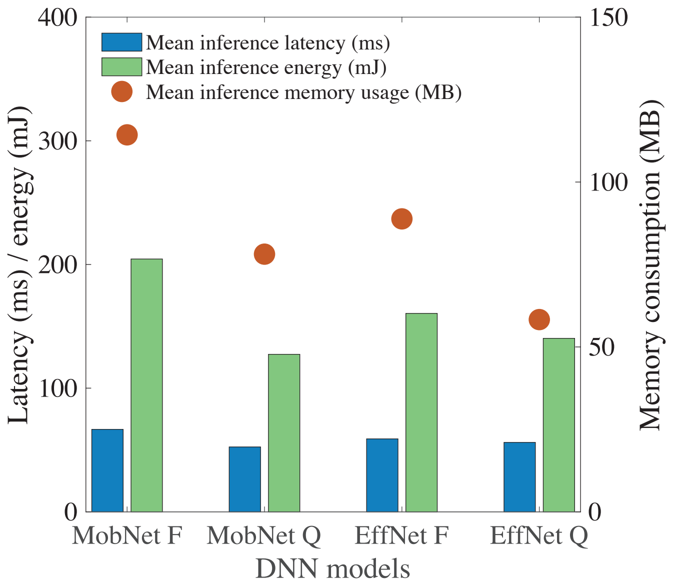


Figure 1.4: Mean inference latency, energy, and memory usage for float and quantized DNN models on Huawei Mate40Pro.

processing sources, size of neural networks, and categories of AI applications (vision and non-vision) [20, 21]. This motivates this research to collect data from a physical testbed to validate this correlation before proposing a predictive energy model [22].

Challenges: Designing a predictive energy consumption model for mobile AI is not straightforward. First, a general energy prediction model is challenging to develop *due to different categorical and numerical variables involved in the non-parametric behavior of the energy consumption of AI applications.* The regression model cannot be linear since all the parameters do not have the same weight in all applications and configurations. Gaussian process regression may provide a solution in this case. Second, *the kernel must be chosen carefully* as there exists a clear link between kernel functions and predictions [23], which contribute to the hyper-parameter optimization. Third, measuring mobile AI parameters is challenging due to *complicated power terminal design in the latest mobile devices.* Synchronizing the timestamps of latency and energy data brings further difficulties as the retrieved log files have different formats. However, these parameters must be measured since they are required for training the regression model. Finally, the *experiments should be controllable and repeatable* for enthusiastic researchers. Therefore, the environment must be chosen wisely so that all the experiments can be carried out in a similar condition.

1.2.2 Performance Analysis Issues of Mobile AI in Heterogeneous Wireless Networks

One of the prominent domains of mobile AI is extended reality (XR) using edge computing. While edge computing can reduce the computational load on XR client devices, it is also imperative to maintain the expected QoS of XR applications, which brings attention to the performance analysis of XR applications. A unified performance analysis model for XR applications is necessary since XR is the next-generation medium for communication, entertainment, manufacturing, defense, automobiles, education, and healthcare [13]. So far, extensive experimental measurement research with a testbed has been proven to be the best way to analyze an XR application's performance [19], which is a time-consuming and laborious process for XR developers and researchers. Moreover, producing datasets of XR applications' performance results is another strenuous task. *A comprehensive and unified performance analy-*

sis model can offer the research community a way to avoid all these exhausting data collection and processing jobs [24].

However, existing research works commonly use single performance metric analysis methods from a macroscopic perspective rather than delving into individual components, which cannot provide a comprehensive insight into an XR service. For instance, end-to-end latency [25, 26] is often considered an important XR performance metric, but this research finds that small segments in the pipeline of running an XR application, such as frame generation, conversion, extractions, and other components, should be analyzed based on their dependencies on hardware settings, the complexity of the task, number of layers in a neural network, and so on. These dependencies influence the XR application's end-to-end latency and energy consumption significantly. Therefore, a comprehensive XR performance analysis model should focus on the individual segments of the application pipeline.

The energy consumption of an XR device depends on the latency of the application, including the computing latency of an edge server [7] and the power consumption during the application run-time. Existing papers propose various ways to model the energy consumption and then minimize the total energy consumption [27–29]. While some researchers break down the total power consumption to hardware levels [30], deeper insights into an XR application's energy consumption can be found in the small segments of an XR pipeline. Apart from these facts, it is well-known that the use of edge computing can lower an XR device's energy consumption via offloading computation- and energy-intensive tasks to the edge. However, if heterogeneous networks and devices are involved in an XR application, the energy model should be inclusive of the latency incurred by different networks, devices, and sensors, which is not considered in existing research.

Sensors and devices of different kinds communicate with an XR device in order to transmit control and environmental information, such as the locations of objects, traffic signals, and street map updates [31]. This communication takes place either directly with the XR device or via an edge server or orchestrator [32]. Additionally, these sensors and devices may be located in the same or different sub-networks using different wireless access technologies such as cellular networks, wireless local area networks (WLAN), and mobile (MANET) or vehicular (VANET) ad-hoc networks [33]. The flow of information through these heterogeneous wireless networks and devices can cause delays in arrival at an XR device, which causes the information to be aged. How frequently the sensors and devices generate the information and transmit it to the XR device is a way to measure the Age-of-Information (AoI), which is an important performance metric for XR applications. Without maintaining the freshness of information, XR users may get the wrong results at the wrong place and time, which eventually can cause a series of effects spreading from nausea to horrible accidents. Furthermore, the mobility of an XR device in heterogeneous networks gives rise to the necessity of both horizontal (within the same sub-network and access technology) and vertical handoffs (different sub-network and/or access technology), which are also known as service migration in the domain of edge computing that plays an important role in both the latency and AoI performance of an XR service.

In this dissertation, it is argued that a comprehensive performance analysis model is necessary for XR applications, which enables researchers to analyze the performance for both local and remote execution of an XR task irrespective of the number or type of sensors or devices and access technologies involved in the service. Consequently, this motivates toward a unified performance analysis model specifically designed for XR applications performing in heterogeneous wireless networks considering latency, energy consumption, and AoI as the performance metrics, which is the first research work on comprehensive XR performance analysis.

Challenges: The latency and energy consumption of an XR device need to be calculated for each individual segment in an XR application pipeline to get a complete insight into the application's performance. The performance of the individual segments depends on the specifications of an XR device, resource allocations, complexity of the task, different parameters set by the applications (e.g., display and encoding), mobility of the device, and wireless channel conditions. Each of these relations is very difficult to present in analytical forms. For example, the computing resource allocation is dictated by the operating system of a device based on priority and parallel operations. In addition, the computational complexity of a task is determined by its type, scope, and goal. Furthermore, the heterogeneity of sensors and networks brings additional challenges to the analytical modeling since it is hard to define each type of sensor and network mathematically without making assumptions that simplify the job but often contradict the field data. Finally, the validation of the model's performance should be done against data collected from experiments with a testbed, which is challenging to design to replicate real-world scenarios.

1.2.3 Service Aggregation in High-Mobility and Low-Latency AI Applications

AoI has been used as a performance metric for low-latency applications, especially in CAVs [34]. It is defined as the time elapsed between information generation by a source and information reception by a requester. It is important for CAVs to maintain a satisfactory AoI to keep track of the road and nearby objects in real-time to ensure public safety as much as possible. Variations in the mobility of vehicles can increase the AoI of the information received, which may fail to meet the application requirements [31]. An initial study shown in Fig. 1.5 indicates that the mean AoI satisfaction rate varies a lot with the relative speed of the ego vehicle with respect to other sensors and vehicles with low coverage areas. AoI satisfaction rate is defined as the percentage of the AoI of the information updates received from stationary sensors and moving vehicles within the upper bound of the required AoI.

In low-latency CAV applications, it is of pivotal importance that the information updates from different sources received by a CAV are sequenced within a short period

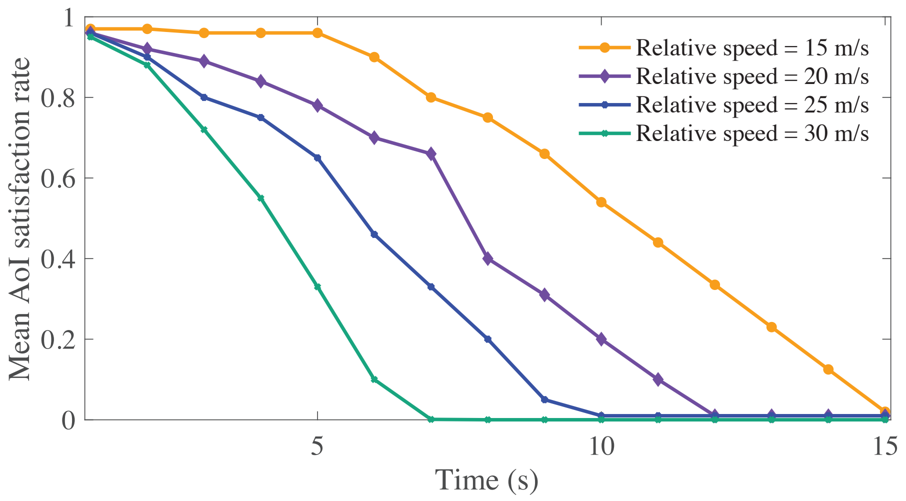


Figure 1.5: Mean AoI satisfaction rate of stationary sensors and moving vehicles at different relative speeds of the ego vehicle.

of time. According to [35], this latency requirement can be as low as 100 ms. If the received information from different sources is not in the correct sequence, or the expected information update is not received at all from a specific source, then the entire application may not function. For example, a pedestrian's exact location is essential for a CAV to determine its next course of action. Another example can be drawn from entertainment applications where out-of-sequence information received from different sources may bring staleness to extended reality services, which in turn will cause discomfort or nausea to the passengers in a CAV.

The high mobility of vehicles and low coverage areas of roadside sensors make the information update service for CAVs more challenging. Existing technologies (e.g., IEEE 802.11p, WAVE, and DSRC) allow the roadside sensors and vehicles to have coverage areas of 100m and 300-500m, respectively [36–38]. The high mobility of vehicles causes serious data sequencing issues. The broadcast messages from the sensors and vehicles received by the ego vehicle need to be processed sequentially and within the required latency. However, *due to the fast movement of the ego vehicle from one coverage area to another, data may arrive out-of-sequence or not arrive at all. In such high mobility scenarios, the ego CAV needs to quickly determine which* *information update service should be terminated, maintained, or established.* In this way, data sequencing can be executed more effectively.

“Service aggregation” in CAVs is defined as processing multiple information update services according to their update cycles sequentially from different sensors and nearby vehicles received by an ego vehicle via broadcast messages [39]. This research considers that service aggregation involves two tasks: one is service connection (i.e., maintain/terminate/initiate a service), and the other is data sequencing. Note that security is an important aspect of service aggregation in CAVs, which is widely studied in wireless sensor networks (WSNs) [40] and VANETs, and not in the scope of this dissertation.

This research argues that AoI is so far used as a performance metric for CAVs, while it has the potential to improve the performance of service aggregation in CAVs as well. The question this work aims to address is, how can the true power of AoI be unleashed (i.e., the use of AoI as a tool to enhance the overall system performance)? This research presents a novel predictive AoI-based service aggregation method for CAVs that can determine highly accurate data processing sequences ahead of time and serve low-latency applications better than existing service aggregation methods in terms of processing delay.

In Fig. 1.6, a simulation of CAVs is demonstrated where the ego vehicle is connected to several roadside sensors and nearby vehicles. The ego vehicle is marked as “EGO”, and other vehicles and sensors are denoted by V and S. In this experimental scenario, sensors and vehicles have coverage areas of 100m and 300m, respectively. This figure also shows the information update buffer of EGO with three distinct update groups at different update cycles. The data denotation in the buffer denotes the information source number and update cycle number (e.g, S_2^3 means the third information update from sensor S_2). In the first scenario, EGO is moving toward the coverage area of S_2 from S_1 and V_2 from V_1. When the relative speed of EGO is 15 m/s, and

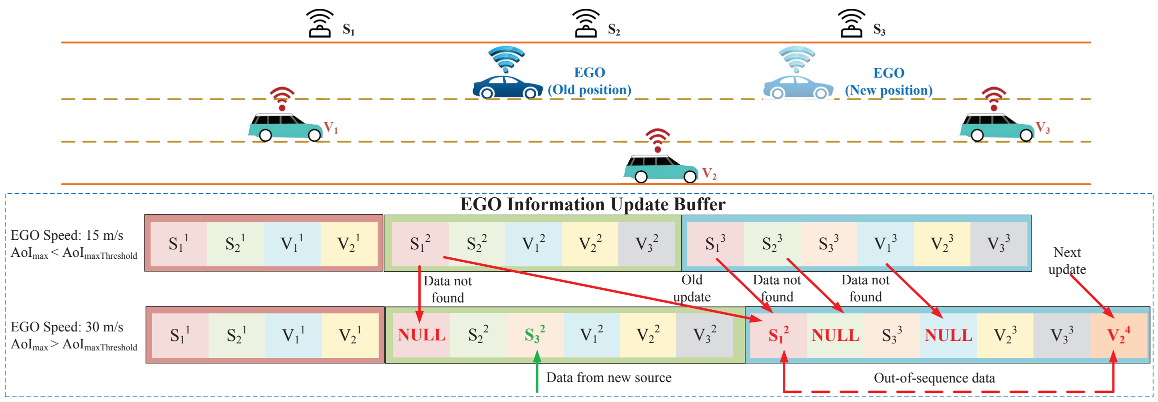


Figure 1.6: Service aggregation problem for varying mobility of the ego vehicle in a CAV scenario.

the maximum AoI (considering all sensors and vehicles) is less than the maximum AoI threshold (required latency), the sequence of the received information update is found to be satisfactory, as shown in the figure. In the other scenario, EGO has a new position at a speed of 30 m/s, where the maximum AoI is higher than the maximum AoI threshold; it is observed that EGO has left the coverage area of S_1 completely. Therefore, the second update from S_1^2 is missing, although the buffer at EGO is expecting it, causing a delay. At the same time, a new sensor's update for the second cycle, S_3^2, arrives at the buffer. At the next update cycle, this becomes even more challenging when EGO leaves the coverage area of S_2 and gains a longer relative distance from V_1. As a result, the expected update from S_1^3, S_2^3, and V_1^3 are missing in the buffer, old update S_1^2 arrives at the buffer, and data from V_2^4 arrives earlier from the fourth cycle due to a lower AoI. This simulation study motivates us to study the impact of relative speed and coverage area on AoI and service aggregation for CAVs.

Challenges: here are several research challenges. First, the sensors and vehicles are heterogeneous in nature, and so are their coverage sizes, which makes it challenging to model the AoI. Second, since service aggregation in CAVs has not been studied before, new performance metrics need to be defined for this specific research. Third, the low-latency applications in CAVs have unique maximum tolerable processing latency requirements, which makes it very challenging to make a service connection decision within the maximum tolerable processing latency. N-step-ahead prediction can be an answer to this problem, but defining N is a challenging task due to the heterogeneity of devices and services. Finally, the prediction will consume the computing resources of the ego vehicle and cost even more time than the processing latency. How to reduce the total number of predictions to save computational resources and time is a crucial factor in such system design. Therefore, this research advocates that the service aggregation system should adopt a periodic AoI prediction policy that requires less latency and computational resources to meet the overall Quality-of-Service (QoS) requirements of the application.

1.2.4 Performance Degradation in Edge-Assisted Mobile AI

Offloading computational tasks to edge servers reduces energy consumption on mobile devices but brings challenges such as increased latency and congested networks. Many resource allocation techniques are proposed that can reduce latency for real-time applications. Nevertheless, mitigating network congestion or alleviating the impact of poor wireless link quality is under-explored for MAR applications. The measurement study presented in this dissertation shows that transmission latency from mobile devices to an edge server increases over 10% and energy consumption rises by around 5% due to a decrease in received signal strength (RSS) as shown in Fig. 1.7. In such a case, reducing the data size through compression can be a potential solution. Moreover, moving object detections in smartphones deal with a high volume of data which can be further reduced using video compression techniques.

To reduce the data size of video frames, compression is a well-researched method [41]. However, compression techniques are mainly studied for powerful machines, not smartphones or other mobile devices. Consequently, there is a need for a trade-off study among latency, energy consumption, data size, and inference accuracy. To

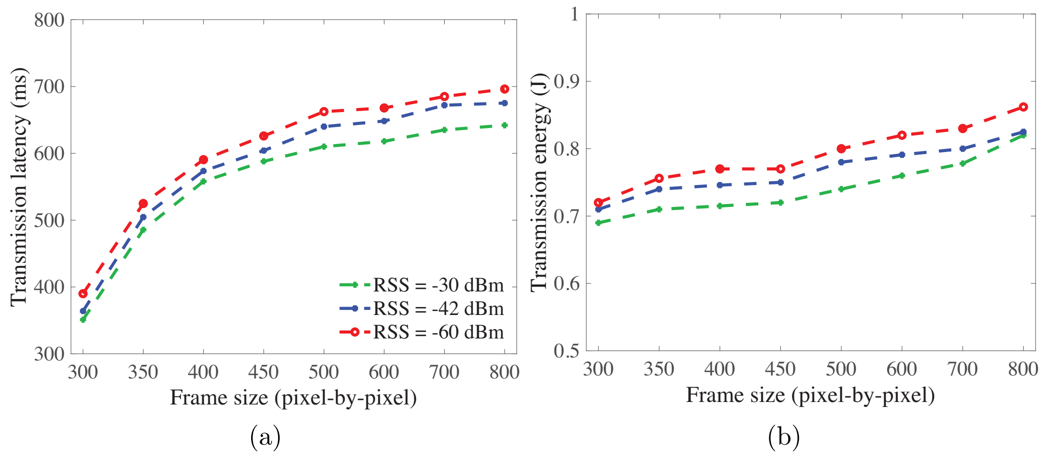


Figure 1.7: Transmission (a) latency and (b) energy consumption at different RSS levels at CPU freq.=1 GHz.

propose a model capable of balancing such trade-offs, how MAR applications behave in terms of latency and energy consumption under different network conditions with or without encoding needs to be understood clearly.

Challenges: The latency and energy consumption of mobile devices in different MAR situations are not linear at all. Through extensive studies, experiments, and measurements, these non-linear traits can be understood. However, setting up an experimental testbed that resembles real-world difficulties involves many challenges.

Moreover, the testbed setup for energy measurement adds further challenges. For Android OS-based smartphones, the energy consumption can be obtained from the on-device log files. However, due to the low sampling rate of this method, it does not provide a precise measurement. Hence, an external power measurement device must be introduced in the testbed that can provide more accurate data with a better sampling rate. However, the input terminal of a smartphone for energy supply is nowadays designed so delicately that accessing these power input terminals and connecting those to an external measuring instrument becomes difficult. Additionally, measuring and collecting the latency data during an MAR activity poses further challenges. While collecting the latency data for each element of the MAR pipeline, each task needs to be organized properly so that the time can be calculated accurately for every event, which is very sensitive in the experiment.

1.2.5 Energy-Intensive Offloading and Data Processing in Edge-Enabled Mobile AI

With the variations in wireless link quality, the communication latency between the edge server and mobile devices may change [19]. Since offloading takes place over the wireless network, the latency rises significantly if the wireless link quality drops. Along with the quality of wireless links, the energy level of mobile devices also plays a vital role in MAR experiences. A battery's energy level or the remaining battery capacity is defined by the difference between its full capacity and the Depth-of-Discharge (DoD), where DoD is simply the percentage of battery capacity used.

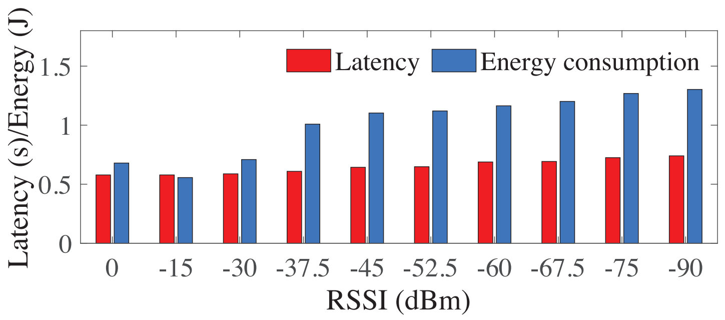


Figure 1.8: Offloading latency and energy consumption of a basic edge-MAR system.

A basic edge-assisted MAR (edge-MAR) application is implemented, and experiments are conducted at different wireless link conditions, where the offloading latency and energy consumption are observed to degrade with the deviation in received signal strength indicator (RSSI), which is shown in Fig. 1.8. Additionally, Fig. 1.9 depicts the energy level of a 3,300 mAh battery in a smartphone executing an object detection model locally and remotely in a good network (RSSI> -30 dBm) and a poor network (RSSI< -75 dBm) using an edge server. This experiment demonstrates that the local execution of a convolutional neural network (CNN) model drains out 90% of the energy in 3.13 hours. Using the remote execution in good network conditions, the battery lasts 4.31 hours, but a poor wireless network causes a 30% reduced battery life. Moreover, current state-of-the-art lithium-ion batteries used in mobile devices are drained at an alarmingly fast rate after a certain DoD is reached [42], which eventually drains the battery long before a user's expectation. Therefore, it is evident that an MAR client will drain its battery much faster due to poor wireless link quality while offloading tasks to an edge server. Needless to say, there is a demand for a smart MAR system that can make offloading decisions dynamically based on the wireless link quality and energy level of an MAR client.

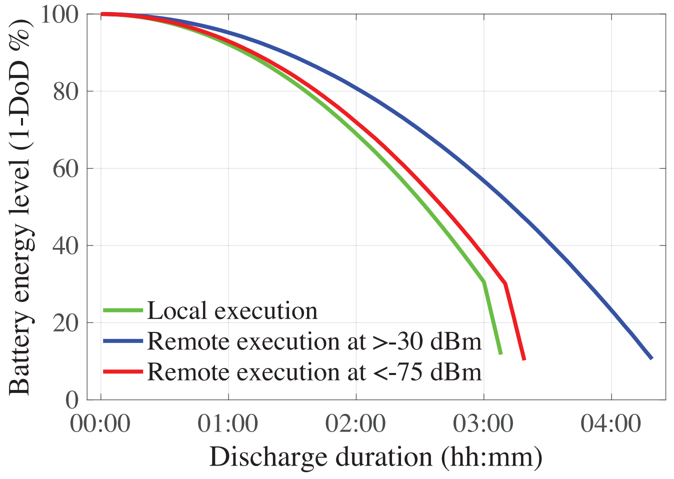


Figure 1.9: Battery discharge of an MAR system on a smartphone.

Besides making offloading decisions, processing input frames is also critical to an MAR system under poor wireless connectivity. The transmitted data size during offloading is determined by how the inputs are processed. Research shows that if the data size gets smaller, the offloading latency goes down [19]. As a result, the effect of poor wireless links on the offloading latency becomes less. Video encoding schemes such as H.264 can compress the input frames and reduce the data size for offloading, lessening the wireless network's burden. However, despite better performances under poor wireless link quality, the energy consumption of mobile devices rises due to encoding. Hence, there is a need to consider the trade-off between latency and energy when deciding whether to process input frames. Therefore, input processing before offloading becomes an integral part of a smart decision system for edge-MAR.

After carefully reviewing state-of-the-art research on edge computing in MAR, three research questions have been identified that are worth exploring.

RQ-1: *How can an edge server intelligently guide the MAR system to reach the optimal point of latency and energy consumption without compromising the QoS?*

Developing a smart edge-MAR system is necessary to defend the QoS against varying wireless link quality, whereas bringing energy efficiency to the system ensures prolonged battery life. The edge server can optimize the performance metrics of an MAR system if all the system states are readily available at any time. However, with the network and energy levels being sets of continuous real numbers, the number of system states becomes infinite with complex state transitions, which makes heuristic optimization algorithms difficult to apply. Moreover, the state-of-health of the batteries and the hardware configurations of mobile devices (e.g., CPU, GPU, and memory) vary over a broad range, which makes it challenging to label the datasets properly. Hence, online recurrent training is needed that enables the system to train itself with the system states on-the-fly and learn from its past experiences. As a result, an online deep reinforcement learning-based (DRL) smart edge-MAR system is proposed — Reinforced Edge-Assisted Learning (REAL), which trains itself from the large set of systems states and takes reward-based actions continuously starting from the initial input.

RQ-2: *How can an edge server provide smart decisions without introducing a significant additional load to the system?*

Introducing learning into an MAR system should be transparent and not hamper the overall QoS. Therefore, the natural choice of placing the learning is on the client side – the mobile device. However, the measurement based on a testbed shows that executing this DRL remotely (at an edge server) can save 22% of the total latency

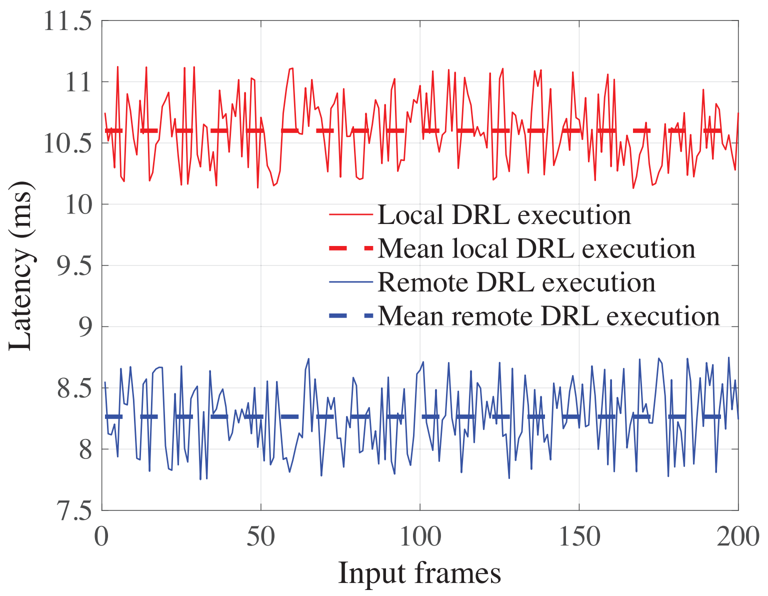


Figure 1.10: Latency for local and remote execution of DRL.

compared to that being executed locally on a mobile device. Fig. 1.10 presents the instantaneous and mean latency of executing DRL (based on a discrete soft actor-critic model) locally and remotely for 200 frames in this experiment. Therefore, this research advocates that the additional load this remote execution of DRL brings forth to the end-to-end latency must be analyzed before adopting the concept of learning at the edge.

RQ-3: *How can the energy consumption and end-to-end latency of an MAR system be obtained accurately under various conditions in order to make smart decisions?*

A smart edge server in an MAR system chooses an appropriate input processing system from multiple probable options for offloading to improve the QoS under various conditions. To achieve this, the energy consumption, latency, and service accuracy should be checked before each decision-making. Since neither analytical models nor measurement data from experiments for energy consumption of an MAR system under different wireless network scenarios are available publicly, physical testbed are needed to collect such data. For Android OS-based mobile devices, the power consumption can be obtained from the current and voltage data by accessing system-level files `cur-`

rent_now and voltage_now from the directory /sys/class/power_supply/battery/ [29], and from battery-level drop data by reading from the Android system variable ACTION_BATTERY_CHANGED [43]. However, due to the low sampling rate of this on-device file-logging method, it does not provide a precise measurement of power consumption. Hence, an external power measurement device is needed in the testbed that can provide more accurate data with a better sampling rate. While an external power measurement device may provide better measurements, it associates with other difficulties. Since the power input terminals of a smartphone for energy supply are nowadays designed delicately to maximize space utilization, accessing these terminals and connecting them to an external measuring instrument is very difficult. These traits of experiments using a testbed make obtaining system states far more challenging.

1.3 Overview of the Proposed Research

The overview of the proposed research is shown in Fig. 1.11. It discusses two broad scopes of the research: one is the performance analysis, and the other is the performance enhancement of mobile AI. The research objectives and work are summarized below.

Research objectives: The objectives of this proposed research are to investigate different mobile AI performance metrics (e.g., latency, energy consumption, service accuracy, offloaded data size, memory consumption, and Age-of-Information) and to formulate mobile AI performance. User mobility in heterogeneous wireless networks is also addressed in this research. The final objective of this research is to design an energy-efficient mobile AI system by mitigating wireless instability effects.

Research work summary: First, this dissertation introduces a measurement study-based performance analysis of mobile AI applications. The latency, energy consumption, and memory utilization are highly impacted by the hardware configurations of a mobile device (e.g., System-on-Chip (SoC), processing sources, number

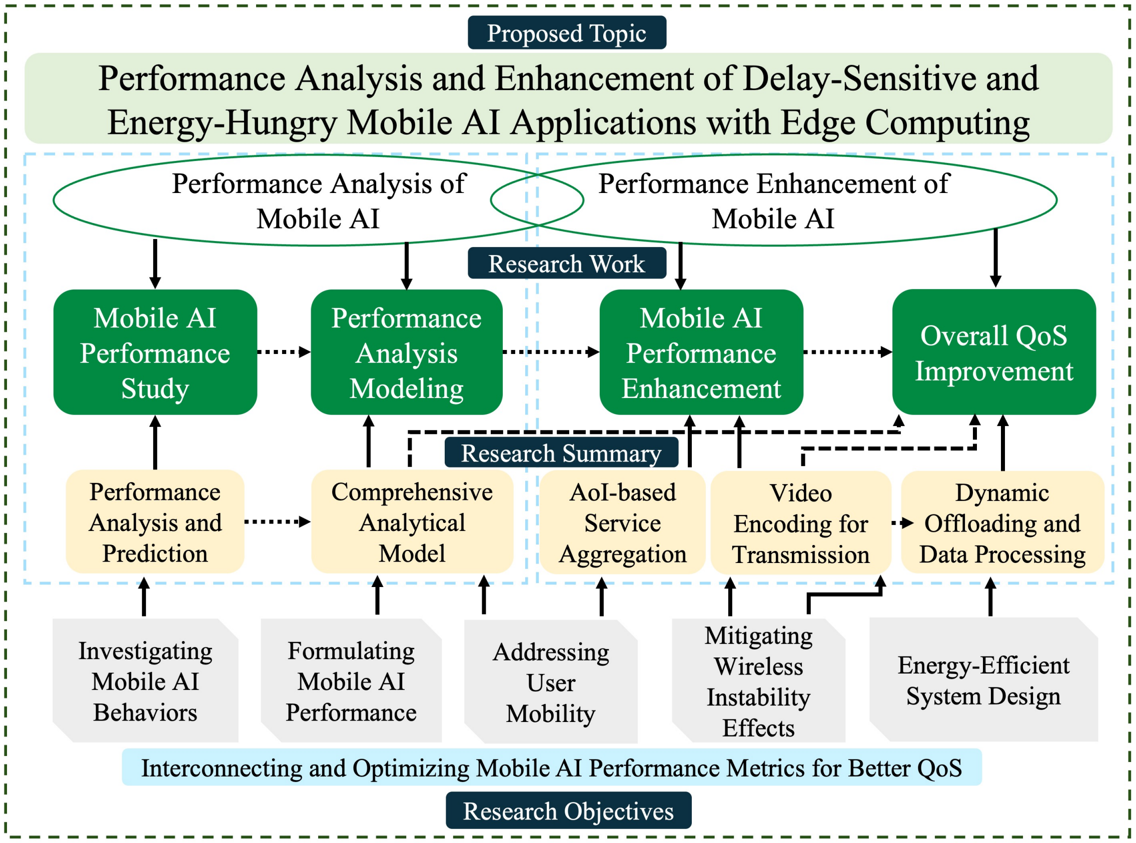

Proposed Topic
Performance Analysis and Enhancement of Delay-Sensitive and Energy-Hungry Mobile AI Applications with Edge Computing
Performance Analysis of Mobile AI
Performance Enhancement of Mobile AI
Research Work
Mobile AI Performance Study
Performance Analysis Modeling
Mobile AI Performance Enhancement
Overall QoS Improvement
Research Summary
Performance Analysis and Prediction
Comprehensive Analytical Model
AoI-based Service Aggregation
Video Encoding for Transmission
Dynamic Offloading and Data Processing
Investigating Mobile AI Behaviors
Formulating Mobile AI Performance
Addressing User Mobility
Mitigating Wireless Instability Effects
Energy-Efficient System Design
Interconnecting and Optimizing Mobile AI Performance Metrics for Better QoS
Research Objectives

Figure 1.11: The overview of the proposed edge-assisted mobile AI performance analysis and enhancement research.

of threads used in processing, number of cores in the processor, and memory size), application type, and the DNN configurations (depth and size of the DNN). After analyzing all these mobile AI behaviors, this research proposes a novel Gaussian Process Regression (GPR) based predictive energy model, which is trained offline, and can predict one-step-ahead end-to-end energy consumption of a mobile AI application. This prediction method is designed to predict the mobile AI energy irrespective of device and DNN configurations.

Second, a comprehensive analytical model for mobile AI (XR) applications' performance analysis in heterogeneous wireless networks is proposed. This research shows the unique challenges of proposing such analytical frameworks and ways to overcome them. Proposing such a unified analytical model for XR performance analysis is not straightforward. The analytical model cannot perform well without considering small individual segments in an XR pipeline since each segment has unique functions and characteristics. Hence, a thorough understanding of each segment is necessary before attempting to model XR performance analytically. Moreover, XR user mobility in heterogeneous wireless networks makes the entire modeling challenging due to vertical handoffs. Additionally, in an XR environment, multiple heterogeneous wireless sensors and devices are considered to transmit control and environmental information to the XR user device. The frequency of information generation by each of these sensors and devices does not follow a similar pattern. This behavior gives rise to analyzing the freshness of information (AoI). Consequently, this research introduces a new performance metric – Relevance-of-Information (RoI). Finally, extensive experimental research has been carried out to strengthen the analytical model with multiple linear regression techniques and also to validate the effectiveness of the proposed model.

Third, since CAVs offer a high-mobility environment for AI applications with low latency requirements, this research delves into solving the service aggregation problem in CAVs. A popular and effective performance metric for CAV applications is AoI, which is used in this research as a tool to improve service aggregation performance. The AoI from heterogeneous sensors and vehicles for the ego vehicle is modeled first. Then, after comparing several prediction models, and based on prediction latency, complexity, and accuracy, the long short-term memory (LSTM) network is chosen for predicting the N-step-ahead AoI. This research also provides guidelines for selecting an appropriate prediction period (N). After that, a novel service connection policy and information update system is proposed for highly accurate service aggregation using the predictive AoI. The proposed system determines when to initiate/terminate/maintain a connection and how to aggregate a service in high-mobility CAV scenarios.

Fourth, this research sheds light on a new aspect of edge-assisted mobile AI applications (MAR) being affected by unstable wireless link qualities. As the received wireless signal strength becomes weaker, the latency and energy consumption of an edge-MAR application go higher. The size of the data to be offloaded to an edge server plays a vital role in this scenario, as the experiment shows that a lower data size can mitigate this challenge posed by the weak RSSI. Therefore, a novel H.264 video encoding-based edge-assisted MAR system is proposed to keep the overall latency and energy consumption of the MAR application even in poor wireless signal strength situations by compressing the offloaded data. However, this encoding has different effects on the individual segments of the MAR pipeline as well as causes a slight degradation in the service accuracy. All these performances are analyzed using experimental data, which has potential future outcomes for the analytical modeling of different MAR parameter designs.

Finally, this research proposes a smart offloading decision and data processing framework for edge-assisted MAR applications to improve the overall Quality-of-Service (QoS). To begin with, the experimental study conducted for this research shows that the discharge pattern of the mobile devices' batteries does not follow a linear trend. With continuously varying wireless link conditions, the batteries discharge even at an alarming rate. Therefore, the edge-MAR system should contain a dynamic decision system based on the system state combinations of wireless signal strength (RSSI) and the remaining battery capacity of the mobile device. However, the complex and huge number of system state combinations make heuristic optimization algorithms difficult to use in this research. As a result, using these two system state information, a reinforcement learning-based solution named the discrete soft-actor-critic method is adopted in the system, which provides actions for each generated input frame. The actions dictate whether to offload and how to process the data. The proposed framework is trained online that runs simultaneously with the MAR application with a view to increasing the overall QoS.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows.

- **Chapter 2:** Related work and novelty of the proposed research are discussed in Chapter 2.

- **Chapter 3:** In Chapter 3, performance analysis and a predictive energy model for mobile AI (EPAM) are presented. EPAM (Energy Prediction for AI in Mobile devices) is designed to predict the end-to-end energy consumption of both vision- and non-vision-based mobile AI application, irrespective of device and neural network configurations. The influencing parameters of EPAM are discussed and the performance evaluation is presented in this chapter.

- **Chapter 4:** A performance analysis framework of XR applications is proposed in Chapter 4. This comprehensive analytical performance analysis model considers local and remote execution of CNNs, user mobility in heterogeneous wireless networks, and communication with heterogeneous external sensors and devices with the XR device to analyze an XR application's performance from the perspectives of latency, energy consumption, and average Age-of-Information (AoI). Finally, the performance evaluation of the proposed framework is presented along with a comparison with state-of-the-art analytical methods.

- **Chapter 5:** A novel service aggregation system for low-latency CAV applications is proposed in Chapter 5. The proposed system provides insight on AoI prediction with different machine learning models and guides the selection of the period of prediction. The service aggregation system is evaluated based on newly introduced performance metrics, and also compared with state-of-the-art data queueing methods.

- **Chapter 6:** In this chapter, a novel H.264 video encoding-based edge-assisted

MAR system is proposed to mitigate the challenges due to unstable wireless networks and to reduce the energy consumption of the mobile device. The performance of the proposed system is studied at different wireless link conditions that resulted into regression-based models to predict MAR performances.

- **Chapter 7:** A smart offloading and data processing dynamic framework for edge-enabled MAR is presented in Chapter 7, which is called REAL (Reinforced Edge Assisted Learning). REAL edge-MAR system is designed to overcome the challenges posed by continuously varying wireless link quality and non-linear discharge properties of the energy storage of mobile devices. Performance evaluation shows that the proposed system "REAL" outperforms other existing offloading methods in terms of the overall QoS.

- **Chapter 8:** Lastly, in Chapter 8, the entire research work is summarized, and future research direction is discussed. Additionally, the publications so far from this research are listed.

CHAPTER 2: RELATED WORK

This chapter discusses the related work in performance analysis and enhancement of different metrics of Mobile AI using edge computing in heterogeneous wireless networks and the novelty of the proposed research work.

2.1 Existing Performance Analysis of Mobile AI

Performance analysis of mobile AI is not a widely researched topic. The parameters on which mobile AI's performance depends need to be analyzed first. Mobile AI applications behave differently in terms of latency and accuracy based on the processing sources [18, 44]. Research works are done on maximizing CPU threads [45] and hardware acceleration [46, 47] for DNN models. The use of GPU is also studied for improving the training and inference time for mobile AI [4]. NPU architectures are explored as well to expedite neural network operations [5, 48, 49]. Moreover, hardware-software co-design methods are proposed to reduce latency in mobile AI applications [50]. Research shows that the memory of mobile devices also impacts the performance of AI applications [51–54]. However, there is no fundamental framework to describe the impact of individual processing sources on energy consumption for different mobile AI applications with disparate DNN models.

Additionally, floating point and quantized models are investigated for vision applications, e.g., image classification, segmentation, super-resolution, and object detection, to create benchmarks using inference latency for mobile devices [44]. Quantized models are introduced in [3, 55–60] to lower the energy consumption and computation complexity as well. Besides, CNN compression is also researched to minimize the energy consumption of mobile devices [61]. In addition, non-vision AI applications are also researched to achieve high accuracy and low latency [62–67]. Nevertheless, a comprehensive performance analysis and predictive energy model for mobile AI requires analysis of complete behaviors of vision and non-vision mobile AI applications using floating point and quantized models, which are not yet explored.

Most existing research works propose performance optimization through analysis models, which are not comprehensive at all [68]. Benchmark suits and testbeds are also developed for such performance analysis of XR – the prominent sub-domain of mobile AI [69]. Some researchers study analysis models from different application perspectives instead of proposing a general one [70]. The performance of AR and VR is analyzed with models for different metrics, such as video bitrate, responsiveness, image noise and rendering, object locations and marker positions, and volumetric data processing performance [71, 72]. However, the scope of these analysis models does not include latency, energy consumption, and AoI, which are truly important in analyzing the performance of XR services and is the key focus of this dissertation.

2.2 Existing Latency and Age-of-Information Analysis of Mobile AI

Latency models for mobile AI and AR/VR applications are presented in several studies [73, 74]. Most of these works focus on communication latency in edge-enabled multimedia services [26, 75], which cannot be applied to either mobile AI or XR applications due to their highly complex operations in computation. In addition, [76] considers the queueing delay of tasks offloaded to an edge server along with the transmission delay, but it considers neither the computation complexity of an XR task nor the handoff. On the other hand, computation latency is considered in several other research works where the computation capability of a device is modeled as cycles [25, 77–79], whereas this research shows that it is a tuple of processing speed (i.e., central processing unit (CPU) frequency), memory size, and available resources determined by the operating system of the device. Furthermore, different approaches toward latency calculation, such as callback functions of applications, are also proposed [80]. Lastly, assumptions are made to model an XR application's latency, such as the use of one single server at a time [1] without considering service migration. This dissertation addresses all of these gaps in a latency analysis model.

In addition, AoI is a relatively newer concept that becomes a performance metric in a mobile AI application when there are several other sensors/devices and network access technologies involved [81]. AoI models are presented in [31, 32], where there are assumptions of packet processing policies and single-access wireless networks. For multi-hop networks, peak AoI is introduced as a performance metric [82], but it does not involve the XR device, which is supposed to receive the information either directly from sensors or from an edge server. Lastly, AoI is used to optimize the overall performance in several research works [83–85], which cannot be applied to general XR scenarios due to the lack of scalability. This dissertation presents a novel AoI model which addresses all of these aforementioned research issues.

2.3 Existing Service Aggregation and AoI in High-Mobility AI Applications

Information fusion or data aggregation is a well-researched topic for WSNs. Numerous information fusion techniques are proposed [86], such as data aggregation based on clustering and compression [87]. However, these methods do not consider the high mobility of CAVs connected to stationary sensors and moving vehicles, which makes it not viable to implement for CAVs.

VANETs and UAVs are also studied in terms of service migration/handoff, data fusion, and task scheduling. Methods for reducing handoff delay or improving the handoff process, and reducing task scheduling load for CAVs and UAVs are proposed [88–90]. Nevertheless, task scheduling often refers to prioritizing tasks on the basis of their importance in a connected vehicular network, which does not solve the problem of aggregation of a singular low-latency service for CAVs.

Furthermore, AoI is recognized as an essential performance metric for CAVs. Researchers propose to minimize AoI through priority-based task scheduling, joint optimization, and machine learning [31,34]. *However, there is a lack of extensive research on the use of AoI to improve the overall system performance.* In [91], AoI is applied to schedule broadcast messages from vehicles to base stations to avoid collisions. Nonetheless, the CAV system considered in this research studies the broadcast service aggregation at a granular level where service connection and aggregation for a single service take place to maintain satisfactory latency using AoI, which has not been researched at all.

2.4 Existing Energy Consumption and Network Resource Utilization Analysis and Measurement of Mobile AI

Energy measurement is necessary to describe mobile AI applications' detailed behaviors. Eprof [92] and E-Tester [93] are proposed to measure and test the battery drain of mobile devices, which use a finite state machine to measure the energy. However, these methods lack in providing granular and precise energy data since they only act on system call traces. Researchers have proposed different energy models for vision [2] and non-vision [94] applications. Furthermore, predictive energy models are developed for devices, and sensors [95]. Nonetheless, developing accurate predictive energy models general to all mobile AI applications requires knowledge of all the environmental parameters such as network and model size, memory usage, and the hardware accessed to run the AI application.

Moreover, energy modeling is a challenging research issue. Analytical models are developed for mobile devices' power consumption in the GPU [96]. In-device logging is often used to measure the power consumption of mobile devices [80], as well as third-party applications such as "Qualcomm's Trepn Power Profiler" [97], "PowerBooter" [98], or "PowerTutor utility" [99], which cannot provide precise measurement data. Unlike these methods, recent research works prefer to use external power consumption measuring instruments [100]. New energy models for mobile devices running deep learning applications for MAR and energy-aware MAR systems are proposed in [2,7] based on measurements provided by an external power measuring tool. Considering all these works together, analytical models and experimental studies are explored on the energy consumption of mobile devices for edge-XR based on input model sizes, CPU frequency scaling, camera sampling rates, input data conversion, and network throughput. However, these works cannot be concluded as comprehensive since each device has its own unique hardware configurations, such as the system-on-chip, memory, CPU, GPU, and AI accelerators. Embracing all these parameters in a generalized model is difficult, but can be achieved by deploying online deep reinforcement learning.

Furthermore, making mobile AI applications network- and energy-aware still remains a research problem. It is shown in [101] that the radio network is accountable for around 33% of the total latency of an edge-MAR system, which infuses a need for network-aware MAR systems. Moreover, preparing an energy-aware MAR system is a dormant research issue since energy modeling is very challenging. Analytical models are developed for smartphones' energy consumption [96]. Some papers use on-device logging to measure the energy consumption of smartphones [80], as well as third-party applications, which do not provide precise measurements. Unlike these methods, recent research works prefer to use external energy consumption measuring instruments [100]. New energy models for MAR and an energy-aware MAR system are proposed in [102, 103]. However, none of the existing systems considers latency and energy consumption along with network resource usage altogether.

Additionally, algorithms are proposed for jointly optimizing configuration adaption and bandwidth allocation for edge-based video analytics systems [104]. Deep neural networks are proposed for runtime optimization in edge-AI devices [105]. Additionally, deep reinforcement learning-based solutions are proposed to allocate resources for edge-assisted applications [106]. Moreover, wireless resource-agnostic XR optimization solutions are provided in [29]. Edge-assisted image quality assessment-based energy-efficient AR is also studied in [107]. Furthermore, computational resource and energy optimization through video encoding is proposed in [19]. On the other hand, sophisticated task scheduling is introduced with a view to bringing energy efficiency in mobile devices [108]. However, all of these optimization works do not fit into the research problem this dissertation addresses since they do not consider poor wireless connectivity with the edge server and rapid battery drainage at low-capacity points of the mobile device. This research addresses these issues while improving the real-time processing of mobile AI applications and the battery health of mobile devices.

2.5 Existing Performance Enhancement of Edge-Assisted Mobile AI

Real-time object detection is one of the main attractions of mobile XR applications [109]. Deep learning networks, more specifically CNNs, are primarily used in detecting objects for increased accuracy, especially in the case of large unstructured training datasets [110]. To get rid of the heavy computational burdens, these applications often exploit edge-assisted services [111, 112]. Edge computing is widely explored for MAR, especially for deep learning-based applications [12, 113]. These works include offloading decisions, resource allocation, energy saving, and service placement [1, 11, 79, 103, 114–119]. Researchers also propose guaranteed offloading in unstable networks and high mobility conditions [120]. Furthermore, DNN partitioning is proposed to deal with varying wireless link conditions [121–123]. Though these research works consider saving computational resources of the mobile devices, none of them investigates the effect of varying link qualities on XR applications and solutions to these challenges, whereas input data processing is proven to be essential to increase the QoS in poor wireless connectivity throughout this research. Data compression may be a key to this problem.

H.264/AVC encoding scheme is a popular standard video coding technology in streaming applications and video file generation [17]. It is also investigated for object detection in video surveillance applications [124]. Existing papers involve the use of H.264 as feature descriptors [125] or for reducing latency [126]. Dynamic video encoding is also used to improve the streaming latency [127]. Nevertheless, none of the existing works describe the changes in MAR behaviors due to encoding in terms of latency and energy, making it difficult to design adaptive systems.

In addition, Reinforcement learning is a popular tool for solving complex research problems. Researchers propose Q-learning and DRL-based task scheduling schemes for edge computing [9, 128, 129]. Additionally, DRL is adopted in [130–132] for video processing, resource allocation, and task offloading in edge computing. However, none of these theoretical works are based on experimental measurements of a mobile device's energy consumption and system latency. The defined states in these works do not represent the whole complex real-world system with uncertainties in the wireless link condition and energy drain of the mobile devices, which are the primary research issues addressed in this dissertation.

CHAPTER 3: PROPOSED MOBILE AI PERFORMANCE STUDY AND PREDICTION

In this chapter, a performance study of mobile AI applications is presented, along with the proposed energy prediction model. An experimental testbed with four different smartphones is set up. A vision application (image classification) and two non-vision applications (NLP and speech recognition) with seven different DNN models are used in the experiment. The testbed is described in detail in Sec. 3.2. Different mobile AI parameters are investigated through an extensive experimental study. The latency, power consumption, and memory usage of individual segments of the pipelines of three AI applications are measured for different applications using single- and multi-threads CPU, GPU, and NNAPI and for different DNN models. The experiment shows that the total energy consumption of a mobile AI application is related to the device configuration, AI model, latency, and memory.

Additionally, a novel Gaussian process regression-based general predictive energy model for mobile AI (EPAM) is proposed based on DNN structure, memory usage, and processing sources to predict the energy consumption of mobile AI applications irrespective of device configurations (Sec. 3.1). EPAM requires offline training with past datasets. The trained model can be used to predict the overall energy consumption, which reduces the necessity for further energy measurement and helps developers design energy-efficient mobile AI applications.

Finally, the performance of "EPAM" is evaluated against experimental data, which shows the credibility of the proposed model (Sec. 3.3).

3.1 EPAM: Overview of the Predictive Model

The energy prediction of mobile AI involves a high dimension of influencing variables, making it a non-parametric model. Let us assume that the set of input data points is $X^{1:D}$, where D is the total number of dimensions. If a noisy observation is considered, then the posterior distribution is found as

$$P(E(X) \propto P(E(X)|\Lambda^{1:D})/P(\Lambda^{1:D}|E(X)), \tag{3.1}$$

where $E(X)$ is the observed energy at data points $X^{1:D}$ and $\Lambda^{1:D} = \{X^{1:D}, E\}$ is observation points. Using Gaussian process [133], $E(X)$ can be described as $E(X) \sim \mathcal{N}(\mu, K)$, where $\mu = [mean(X^1), \ldots, mean(X^D)]$ is the mean and $K_{ij} = k(x_i, x_j)$ is the covariance or Kernel function, where x_i and x_j are distinct data points.

As new data points X_* are provided, the posterior distribution of predicted energy $E(X_*)$ can be modeled as

$$P(E(X_*)|\Lambda^{1:D}) \sim \mathcal{N}(\mu(X_*), K(X_*)) \tag{3.2}$$

The kernel must be chosen carefully as there exists a clear link between kernel functions and predictions [23], which contribute to the hyper-parameter optimization. From the experimental data, it is observed that *the influencing parameters on total energy consumption are sparse and vary over a broad range including both numerical and categorical variables.* Hence, the automatic relevance determination (ARD) exponential squared kernel is chosen for the predictive model, which automatically puts different weights on the parameters with differential scales assessing their significance to the model. Hence, the kernel equation becomes:

$$K(x_i, x_j) = \sigma_f^2 \exp[(-\frac{1}{2}) \sum_{m=1}^{D} \frac{(x_{im} - x_{jm})^2}{\sigma_m^2}], \tag{3.3}$$

where σ_f^2 is the hyper-parameter to be optimized and σ_m^2 is the covariance of the m^{th} dimension. Finally, the log-likelihood of the trained model can be expressed as

$$\begin{aligned}\log P(E(X)|X^{1:D}) = &-\frac{1}{2}E(X)^T(K+\sigma_D^2 I)^{-1}E(X) \\ &-\frac{1}{2}\log det(K+\sigma_D^2 I) - \frac{D}{2}\log 2\pi,\end{aligned} \tag{3.4}$$

where I is an identity matrix. EPAM is first trained offline with the observation data points, then is run with an application alongside. The prediction is done either simultaneously or at the end of an application. In this research, the model is trained with a dataset containing $85,500$ data, validate with $19,496$, and test with $10,000$ data.

3.2 Experimental Setup

In this section, the application pipelines considered for this research, testbed, AI models, and performance metrics are discussed in detail.

3.2.1 Pipelines of AI applications

Three mobile AI applications are used in this research: image classification, NLP, and speech recognition. The pipelines of these applications are described below.

3.2.1.1 Image Classification

In image classification, as shown in Fig. 3.1, first, the image is captured by the camera sensor, which then goes through a Bayer filter and image signal processor, and then is stored in an image buffer. The image frame is then scaled and cropped to be previewed while simultaneously going to an image reader, converted from YUV color format to RGB, and cropped according to the input size of the DNN model. Then the converted and cropped frame is taken as the DNN input, generating the classification results to display.

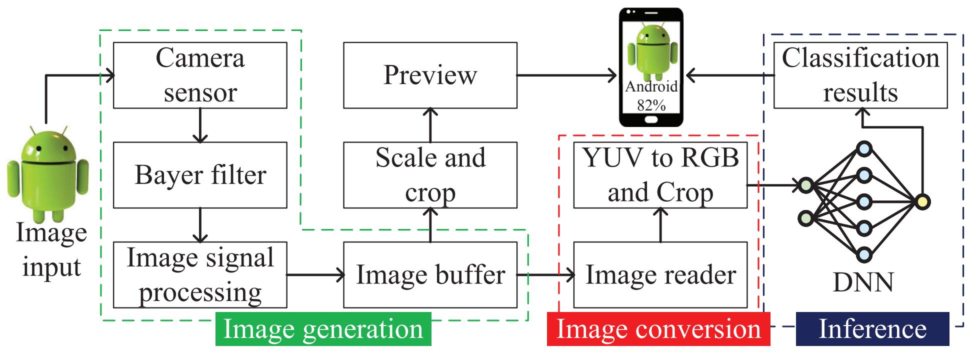


Figure 3.1: Pipeline of image classification application.

3.2.1.2 Natural Language Processing (NLP)

The NLP question-answer application takes both the paragraph input and the question input from the keyboard (Fig. 3.2. The paragraph is then represented with token, segment, and position embeddings. The keyboard input goes through character, basic, and word piece tokenizer. These embeddings and tokens are passed to a feature converter providing input to the DNN model. The model finds the answer to the question input and highlights it in the paragraph.

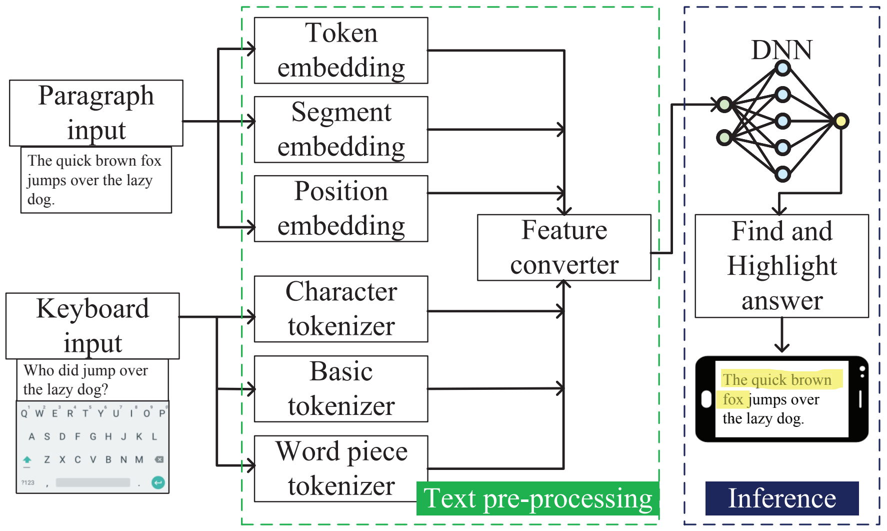


Figure 3.2: Pipeline of NLP application.

3.2.1.3 Speech Recognition

Speech recognition application records, converts, and decodes the audio input. The decoded audio signal is converted to a spectrogram by running a short-time Fourier transform (STFT) along with the calculation of the Mel frequency cepstral coefficients (MFCCs). The spectrogram and MFCC are passed to the DNN model. The predicted word is then displayed on the phone as depicted in Fig. 3.3.

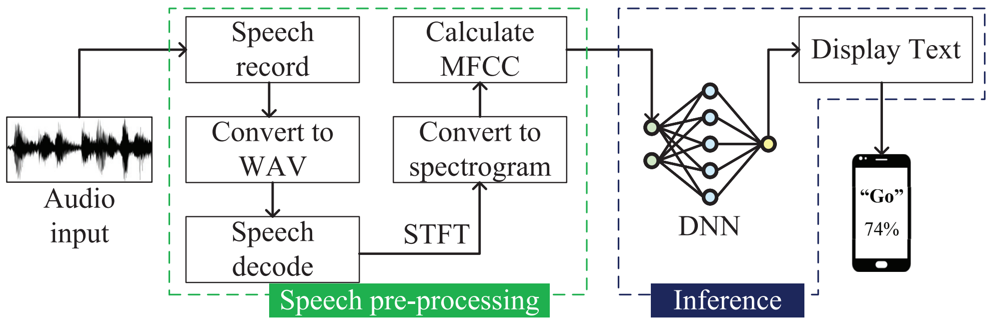


Figure 3.3: Pipeline of speech recognition application.

3.2.2 Testbed

The applications mentioned above are implemented on four Android OS-based smartphones from different manufacturers with distinct configurations to make the measurement study robust with a wide range of parameters. Table 3.1 shows the specifications of the smartphones used in the experiment. However, the intended thorough investigation of mobile AI brings several challenges during the experiment.

Challenge #1 – *How to measure the power and memory usage of each segment of an AI pipeline?*

Android Studio, along with other third-party contributors, provides developers with memory and battery profilers, which cannot generate the data necessary to measure memory usage and power consumption precisely. In this experiment, latency timestamp data of each segment of a mobile AI pipeline are collected along with their corresponding memory usage. To measure energy consumption, an external power

Table 3.1: Brief specifications of the devices used in the experiments for mobile AI performance study and prediction

Denotation	**Device-1**	**Device-2**	**Device-3**	**Device-4**
Model	Huawei Mate 40Pro	One Plus 8Pro	Motorla One Macro	Xiaomi Redmi Note8
SoC	Kirin 9000 (5 nm)	Snapdragon 865 (7 nm)	Helio P70 (12 nm)	Snapdragon 665 (11 nm)
CPU	Octa-core (1 × 3.13GHz A77 3 × 2.54GHz A77 4 × 2.05GHz A55)	Octa-core (1 × 2.84GHz 3 × 2.42GHz 4 × 1.8GHz Kryo 585)	Octa-core (4 × 2GHz A73 4 × 2GHz A53)	Octa-core (4 × 2GHz Gold 4 × 1.8GHz Silver Kryo260)
GPU	Mali G78	Adreno 650	Mali G72	Adreno 610
Dedicated AI accelerator	Ascend Lite+ Tiny NPU Da Vinci	Hexagon 698 DSP	MediaTek APU	Hexagon 686 DSP
RAM	8 GB LPDDR5	8 GB LPDDR5	4 GB LPDDR4X	4 GB LPDDR4X
OS	Android 10	Android 10	Android 9	Android 10
NNAPI Support	Yes	Yes	Yes	Yes
Release Date	October, 2020	April, 2020	October, 2019	August, 2020

measurement tool "Monsoon Power Monitor" is used that provides data sampled at every 0.2 ms interval. However, the latest smartphones' power input terminals are difficult to find since they are delicately connected to non-removable batteries. Therefore, the devices need to be heated and opened to remove the battery and then connected to the power monitor. After careful measurement of power data, they are matched with the corresponding latency timestamps.

Challenge #2 – *How to make the experiment environment controlled?*

To make the experiment environment controllable, all the experiments are carried out in a similar condition, e.g., brightness, camera focus, image resolution, background applications, processing sources, and test dataset. For this experiment, 640×480 pixels are used as the image resolution, and `TensorFlow Lite Delegate` to control the processing sources. The 2017 COCO test dataset, WH-questions, and fixed single words are used for testing the classification, NLP, and speech recognition, respectively.

Challenge #3 – *How to distinguish the power required by mobile AI from the base power consumption of a mobile device?*

Mobile devices start consuming power right from it is powered up. Even without any applications running in the background, there is always a minimal power consumption – which is called the *base power* in this research. To distinguish the mobile AI power from the base power, an additional layer is used before the actual AI application – assuring correct measurements.

3.2.3 AI models

In this research, seven DNN models are used for three different applications. In Table 3.2, the details of each model, including the input size, number of layers, and the trained model size (occupied storage space), are shown.

3.2.4 Performance metric

All the AI applications' performances are analyzed in terms of their latency, energy consumption, and memory usage. The total energy consumption is controlled by latency and memory usage, as well as the category of AI applications, processing sources, model types (float and quantized), and DNN structure and model size.

3.3 Mobile AI Performance Analysis

Experiments are conducted with all the devices listed in Table 3.1 and models listed in Table 3.2 by switching to different processing sources, such as CPU thread 1 and

Table 3.2: DNN models used in mobile AI performance study

Denotation	**Model**	**Appliation**	**Input size**	**No. of layers**	**Model Size**
Model 1	MobileNeV1 (Float)	Image classification	224x224x3	31	16.9 MB
Model 2	MobileNetV1 (Quantized)	Image classification	224x224x3	31	4.3 MB
Model 3	EfficientNet-lite (Float)	Image classification	224x224x3	62	18.6 MB
Model 4	EfficientNet-lite (Quantized)	Image classification	224x224x3	65	5.4 MB
Model 5	NASNet Mobile (Float)	Image classification	224x224x3	663	21.4 MB
Model 6	Mobile BERT QA	NLP	int32 [1, 384]	2541	100.7 MB
Model 7	Tensorflow ASR	Speech recognition	[20 Hz, 4 kHz]	8	3.8 MB

thread 4, GPU, and NNAPI. Models 1 to 5 are for vision-based AI, and models 6 and 7 are for non-vision-based AI applications. It is to be noted that models 2, 4, 6, and 7 do not support GPU processing due to a lack of `TensorFlow Lite` optimization. In general, the applications have input data processing (combining image generation and conversion in classification) and inference tasks. Some of the interesting findings from the analysis are shown in this chapter.

3.3.1 Latency and energy consumption of mobile AI

The end-to-end latency and energy consumption per cycle for all the models with different processing sources are shown in Fig. 3.4. First, it is observed that quantized models decrease the inference latency (13%) and energy consumption (25%) from their respective float models. Additionally, there is a reduction in the overall latency of 4% when switching to a 4-thread from a single-thread CPU. However, in quantized models, the multi-thread CPU processing slightly increases the total energy consumption (3% on average). The use of GPU even lowers the end-to-end latency and energy consumption compared to the use of single-thread CPU (8% and 27% respectively on average) and 4-thread CPU (7% and 25% respectively on average). On the contrary, NNAPI behaves differently than the other three processing sources on different devices. First of all, a reduction is observed in overall latency of 19% from model 1 to model 2, which is a quantized version of model 1, and a 7% reduction from model 3 to model 4. For models 4 and 5, NNAPI increases latency and energy considerably. The insight here is that NNAPI can perform better with sufficient hardware support from the manufacturers.

An interesting fact about the NLP application is that the text processing step shows an entirely different latency pattern. This segment takes user input which does not take uniform time, i.e., it varies with user habits of typing and thinking of the question. Moreover, NLP processes the keywords based on the structure of the sentences and punctuation marks from the input. Hence, the processing stage here is completely unpredictable for different users. In NLP, each input consumes around 5.7 J, whereas, another non-vision application, speech recognition takes around 161.85 mJ to process one speech input sampled at a rate of 16 kHz using a single-thread CPU. NNAPI consumes the least latency and energy for speech recognition.

In addition, the power consumption charts of different applications and processing sources are also analyzed (Fig. 3.5). It is observed that a slight initiation delay occurs for every application (marked with red arrows in Fig. 3.5), which varies with using different processors and applications. This delay occurs during the time when the application interface initiates till the activity-start point, which is mainly originated by different hardware components being accessed at the beginning of an AI application, such as the camera, keyboard, speaker, and microphone. Besides, different processor delegations (e.g., GPU and NNAPI) are also done during this period.

From this observation, the energy consumption of a complete application cycle can be modeled as:

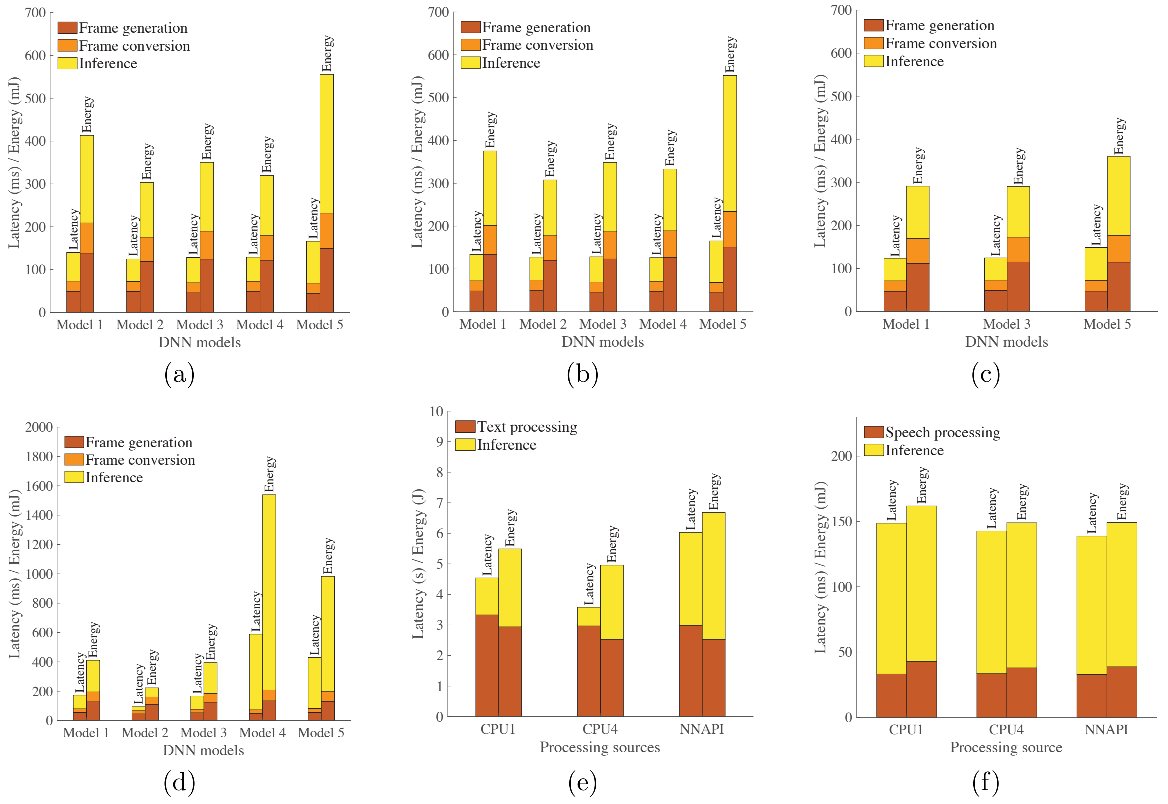
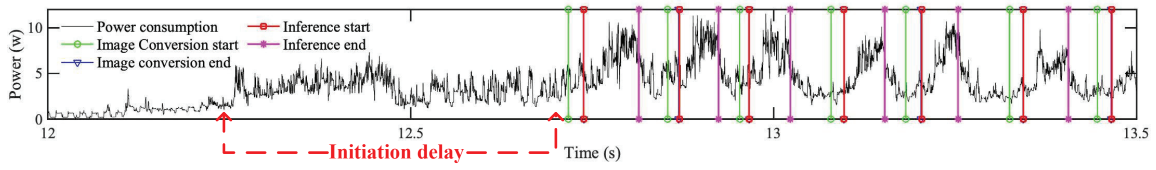


Figure 3.4: End-to-end mean latency and energy consumption per cycle of vision-based models 1–5 for (a) single- and (b) multi-thread CPU, (c) GPU, and (d) NNAPI, and non-vision-based (e) model 6 and (f) model 7.

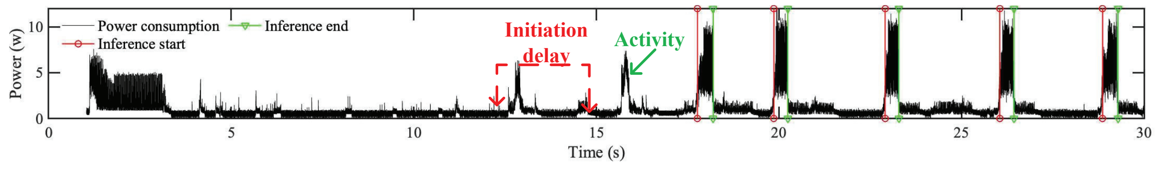
$$E_{total} = E_{id} + n(E_{proc} + E_{inf}) \tag{3.5}$$

where E_{total} is the total energy consumption during an application cycle, E_{id} is the energy consumption during initiation delay, n is the total number of inputs processed (e.g., image, text, or audio), E_{proc} is the mean energy during input processing (for vision applications: image generation and conversion), and E_{inf} is the energy during inference per input.

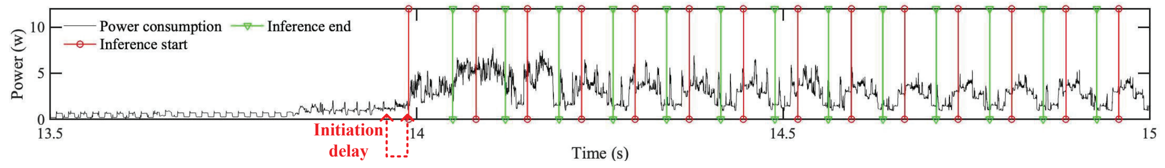
Highlights: *Non-vision applications cannot be generalized for latency and energy like vision-based ones. GPU processing is not supported by non-vision applications, which should be explored widely. The initiation delay (i.e., the delay between the activity trigger and start point) varies along AI models, processing sources, and ap-*



(a)



(b)



(c)

Figure 3.5: Power consumption pattern for (a) classification, (b) NLP, and (c) speech recognition.

plications, which is caused by accessing different hardware components by mobile AI applications.

3.3.2 DNN structures and their inference latency and energy

DNN structures define the way inference activities work in a mobile AI application. The behavior of DNN structures varies across different kinds of applications as well, e.g., vision and non-vision AI. For instance, a smaller DNN structure for vision applications can incur higher latency and energy than a larger non-vision DNN structure. Inference latency and energy consumption per cycle are shown in Fig. 3.6 for DNN models with single-thread CPU processing. It is observed that model 5 takes longer inference time and energy due to its larger structure than the other vision-based AI models. The longest latency and highest energy are evident in model 6 (a complex structure comprising 2541 layers).

Highlights: *DNN structures influence inference latency and energy significantly, but the relationship is not linear at all. Generally, larger DNN structures are respon-*

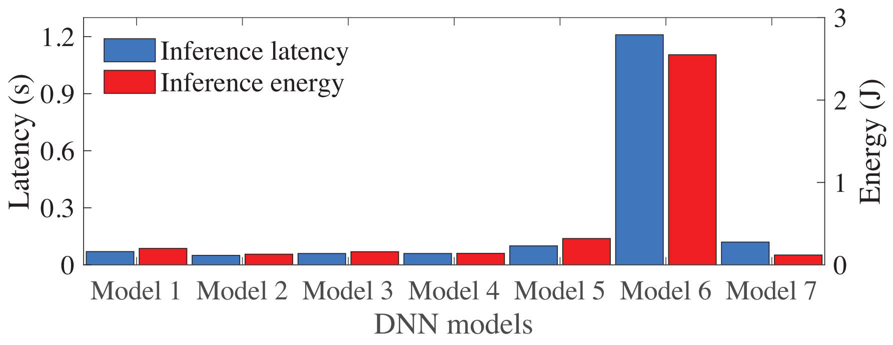


Figure 3.6: Inference latency and energy consumption per cycle by DNN models.

sible for higher latency and energy for a mobile AI application.

3.3.3 DNN model size, memory usage, and inference energy

DNN model size (i.e., the storage space occupied by the model) impacts memory usage and energy consumption during inference. From the experiment, model 7 is observed to have the lowest model size, hence causing the lowest memory and energy consumption, whereas model 6 has the highest size, memory, and energy consumption. This is more evident from Fig. 3.7, which shows a comparison among all the models' sizes, inference memory, and energy consumption for single-thread CPU processing.

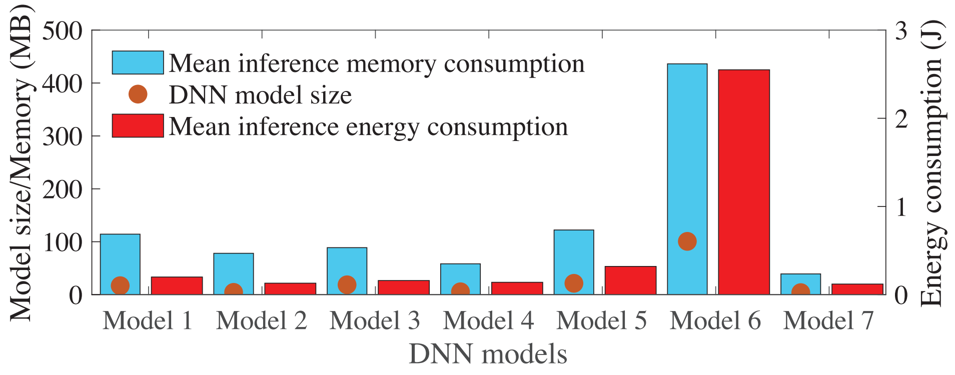


Figure 3.7: Comparison of DNN model size, inference memory usage, and inference energy consumption.

Additionally, In Table 3.3, the memory usage due to inference is shown for different configurations. The slightest memory is used by CPU thread 4 for models 1 and 2, and GPU for models 3 and 5. For model 4, CPU thread 1 provides the best performance

Table 3.3: Memory usage by different DNN models with different processing sources

Models	**Processors**	**Mean memory consumption due to inference (MB)**
Model 1	CPU1	114.314
	CPU4	82.8112
	GPU	92.698
	NNAPI	101.430
Model 2	CPU1	78.131
	CPU4	76.344
	NNAPI	128.020
Model 3	CPU1	88.828
	CPU4	96.722
	GPU	81.539
	NNAPI	119.844
Model 4	CPU1	58.291
	CPU4	76.376
	NNAPI	110.299
Model 5	CPU1	122.243
	CPU4	118.424
	GPU	114.281
	NNAPI	139.072
Model 6	CPU1	436.276
	CPU4	428.221
	NNAPI	91.7534
Model 7	CPU1	39.230

since its quantization impacts the rich processors.

Highlights: *Lower memory used by mobile AI applications ensures computation resources and energy for other mobile device activities. From this perspective, quantized and smaller DNN models are best suited for mobile AI. The larger the storage occupied by a DNN model, the higher the memory and energy consumption.*

3.3.4 Performance evaluation of EPAM

The Gaussian process regression-based predictive energy model, EPAM, is developed and trained with each device's SoC, CPU frequency, no. of cores, memory size, processing sources, no. of threads, application type, DNN model, DNN structure, memory usage, processing latency, and inference latency from the large experimental dataset from this research to predict the total energy consumption per application cycle (data processing and inference for each input). An empty basis function and

ARD squared exponential kernel function are used for the hyper-parameter optimization. We use device-1, 2, and 4 for training and validation, and device-3 for 1-step ahead prediction testing. In this section, only a few prediction results are shown in Fig. 3.8. EPAM's energy prediction per cycle is highly accurate for all the models. The overall root mean squared error (RMSE) is 0.075 (3.06%), and the marginal log-likelihood value is -1.449×10^2, which show that the trained model is a good fit for the prediction. The prediction latency depends on the machine used to run the model.

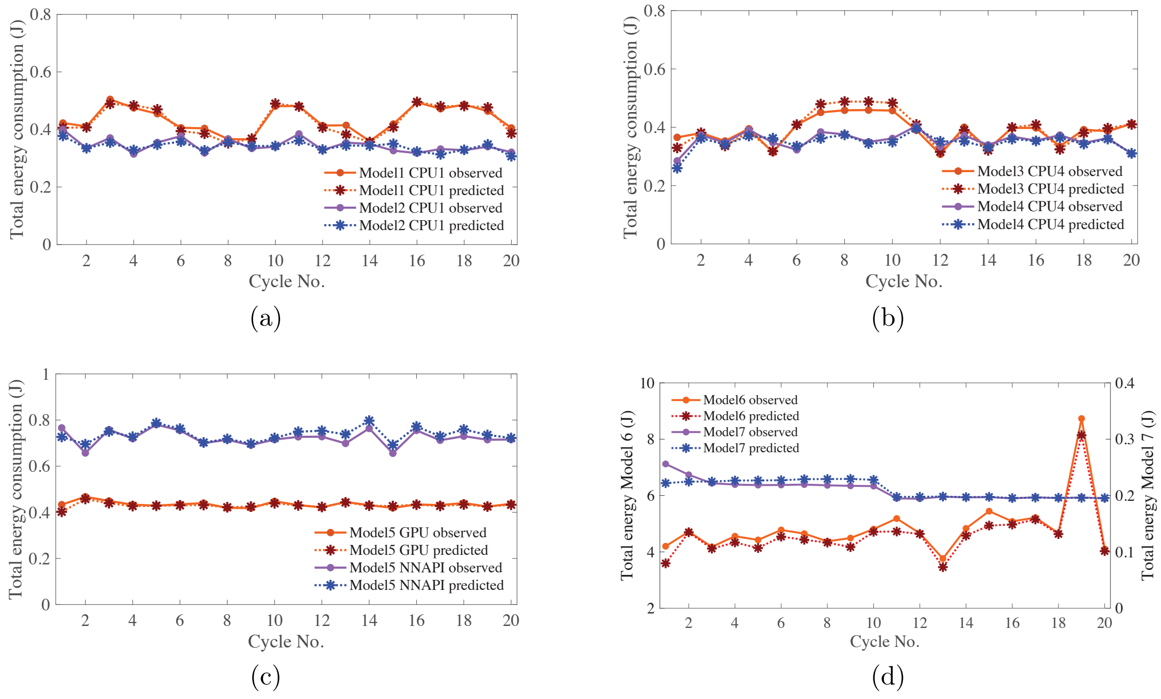


Figure 3.8: Evaluation of EPAM for (a) model 1 and 2 (b) model 3 and 4, (c) model 5, and (d) model 6 and 7 with different processing sources.

Highlights: *EPAM further helps developers and users to perceive the performance of individual AI applications in terms of energy with high accuracy – which is the primary motivation of this research work. The larger and more diverse the training dataset, the higher the prediction accuracy.*

CHAPTER 4: PROPOSED PERFORMANCE ANALYSIS FRAMEWORK FOR MOBILE AI

In this chapter, a performance analysis framework is proposed for XR applications performing in heterogeneous networks. First, the system model, along with the considered XR application pipeline, is discussed in Sec. 4.1. Then, the proposed modeling of the key performance metrics, namely, latency, energy, and AoI in Sections 4.2, 4.3, and 4.4, respectively, along with associated challenges and mitigation techniques are presented.

Following this, the experimental research is discussed, which is conducted on an XR testbed to create a dataset with a view to validating the proposed model and strengthening the model where regression is needed to avoid unrealistic assumptions. The methodology and experimental setup are discussed in Sec. 4.5.

Lastly, the performance of the proposed analytical framework is evaluated in terms of latency, energy, and AoI with respect to real experimental data in Sec. 4.6. The framework is also compared to other state-of-the-art analytical methods, where the proposed framework outperformed every other methods.

4.1 Overview of the Proposed XR Performance Analysis Modeling Framework

XR applications are diverse in operation and scope. Ranging from virtual reality games to futuristic infotainment systems in autonomous vehicles, XR applications can have different pipelines and features of operation. Therefore, it is difficult to devise a general performance analysis model for XR. To tackle this challenge, the operation and pipeline of the application need to be understood in detail first. This knowledge of the pipeline can further lead to the analysis modeling. The proposed XR performance analysis framework provides a guideline for enthusiastic researchers in this domain on how to model different performance metrics of an XR application. The performance metrics studied in this research are end-to-end latency, energy consumption, and AoI. This research shows that the following steps are essentially helpful in designing such analysis models.

- First, the XR application pipeline, including the individual segments, needs to be identified.
- Second, the operation of the pipeline needs to be studied. For example, the segments operating sequentially and in parallel need to be distinguished. Moreover, the operations in a data buffer need to be considered in such modeling.
- Third, the factors influencing the latency and energy of each segment need to be analyzed. For example, the latency for frame generation depends on the frame rate, frame resolution, data size, and the allocated computing resource and memory bandwidth of the device. Other segments' details are also discussed in later sections.
- Fourth, when explicit analytical modeling form is not possible for some segments, other numerical methods, such as regression analysis, are required to find the analytical form. For example, the encoding and decoding latency depends on so many factors (e.g., different configuration parameters) that a direct analytic form is very hard to find.
- Fifth, the analytical modeling for AoI of an XR application needs in-depth knowledge of sensors and devices involved in the pipeline and their communication methods. The packet arrival and service rates, propagation models, and path loss models need to be considered if applicable.

- Finally, the models' validation needs to be done with appropriate data, preferably collected through experiments. This validation enables researchers to find flaws in the models and fix them accordingly.

XR applications have complicated pipelines since they contain the attributes of AR, MR, and VR applications. In this research, the basic components of an XR application have been identified, which can explain any other XR application with slight modifications. For example, a multiplayer XR game can include database sharing [134], and a vehicular application can consider map data sharing with the cloud and other vehicles – which are based on the example XR pipeline explained below.

To illustrate the proposed modeling framework, in this section, the entire pipeline of an XR application (object detection) is broken down into segments that have unique functions and responsibilities. The segments having parallel operations are considered accordingly in the analysis modeling framework. Fig. 4.1 shows these individual segments and an overview of their functions.

- **Frame generation:** The XR device captures a frame from the real world using the camera sensors at a predefined rate (i.e., frame rate). Then, the captured frame is processed by a Bayer filter and signal processor to prepare it for the *input buffer*.

- **Volumetric data generation:** The XR device calculates the inertial data of the user (i.e., the person wearing the head-mounted device (HMD) or an autonomous driving system (ADS)). The inertial data is used in the 6 Degrees-of-Freedom (DoF) localization, which is necessary to understand the user's exact location in a three-dimensional (3D) space. Lastly, the 3D point cloud data is extracted from the current scene the XR device is displaying. This data is passed to the *input buffer*.

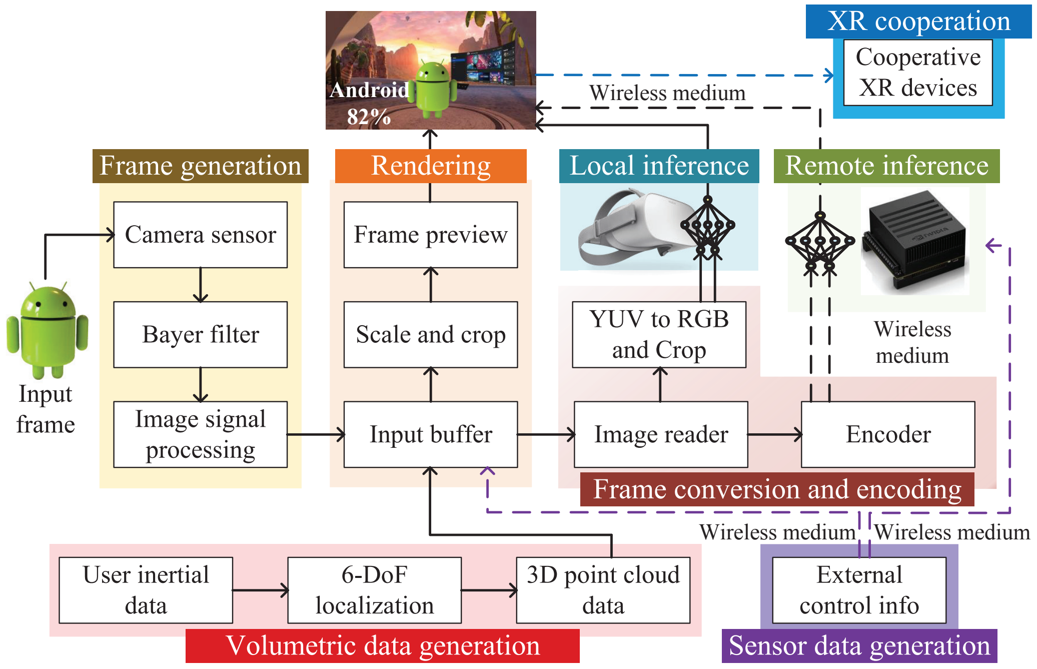


Figure 4.1: XR application pipeline for object detection.

- **External sensor information generation:** In this research, external sensors and devices are considered to be involved in the XR application. The sensors or devices generate and/or provide control information, such as the location of an object or a person in a scene (e.g., a virtual meeting, multiplayer games, and pedestrian location in an ADS). This information is transmitted to the XR device via a wireless medium and stored in the *input buffer* along with the generated frame and volumetric data.

- **Frame conversion:** This segment is necessary for the *local inference* (i.e., on-device inference with a light-weight convolutional neural network (CNN)). The raw frames captured in *frame generation* are generally in the color format YUV, which needs to be converted to RGB to match the input format of the CNN. Moreover, the input tensor of the CNN has specific frame resolution requirements which are met through scaling and cropping in this segment. An *image reader* reads the frames from the input buffer and passes them to the *frame converter.*

- **Frame encoding:** This segment is related to the *remote inference*. *Frame encoding* compresses the size of the data to be transmitted to an edge server by using predictions and removing redundant features in a frame. In this research, the standard H.264 is considered for the encoding and decoding of the frames. The encoding quality depends on several parameters such as the intervals of I-, P-, and B-frames, bitrate, frame size (i.e., resolution), frame rate (i.e., frames per second (fps)), and quantization value.

- **Local inference:** The XR device can detect objects using an on-device lightweight CNN (i.e., CNN with a low number of layers and size), which is used to calculate the inference results without incurring a high delay and energy consumption since the device has a limited computation capability and battery backup. The input from *frame conversion* is directly fed to the *local inference*, and the output of this segment is passed to the *frame renderer.*

- **Remote inference:** If the XR device decides to calculate the inference results on an edge server, the *remote inference* is used. In this segment, the encoded frames, along with the volumetric data and control information, are passed to the remote edge server via a wireless medium, which has higher computation resources and a larger CNN to provide more accurate and faster results. The edge server decodes the frame and generates the inference result. The result is then passed to the *frame renderer* via the wireless medium. Note that the computation task can also be distributed between the XR device and edge server(s) (one or multiple servers), based on the application's requirement.

- **Frame rendering:** The volumetric data, captured frame, and external control information are used in *frame rendering.* The captured frame is scaled and cropped as per the display requirement of the XR device. In this segment, all of these inputs to the *frame preview* are processed to be displayed accordingly. Finally, the inference results, from either the XR device or the edge server, are passed to the *frame renderer* to display the results alongside.

- **XR cooperation:** The XR device may communicate with other cooperative or collaborative XR devices through edge or cloud servers via a wireless medium. This scenario is common in multi-player gaming or cooperative XR applications. In this segment, the XR device sends either the entire scene or fragments of information to convey specific objects' position and alignment. Generally, this segment is executed alongside frame rendering, hence it is dependent on the application whether to include XR cooperation to the end-to-end latency and energy calculation.

4.2 The End-to-End Latency Analysis Model

4.2.1 Challenges in Latency Modeling

An XR application generally has two kinds of latency: computation and communication latency [1]. Due to the unique nature and functions of each individual segment of an XR pipeline, analyzing the latency for each of the segments can provide a better insight into the application [2]. However, the latency of each component depends on several other distinct device and network parameters, which brings additional challenges. For example, most of the computation segments depend on the available computation resources, which cannot be modeled simply. This is because, today's XR devices and edge servers are equipped with modern GPUs. Moreover, the use of tensor processing units (TPUs) and neural processing units (NPUs) is also on the rise. The XR application and the OS of each device together determine which processing units to use and at what utilization ratio.

Furthermore, research shows that the depth and size of neural networks (NNs) have impacts on the latency [22], but the modeling of such impact is not provided in existing works. Exploring the relationship between NNs and the corresponding latency is challenging since each NN is unique in nature in terms of the interconnectivity and use of layers. These challenges are addressed and solved in the following subsection.

4.2.2 The Proposed Latency Analysis Model

T proposed latency analysis model consists of the latency from each segment of the XR pipeline for each generated frame. For instance, consider the latency for the q-th frame, where $q \in \{1, 2, ...Q_n\}$ such that Q_n is the final frame at the end of the application. The end-to-end latency of an XR application can be expressed as

$$\begin{aligned} L_{tot}^q = & L_{fg}^q + L_{vol}^q + L_{ext}^q + L_{ren}^q + \omega_{loc} L_{fc}^q \\ & + \bar{\omega}_{loc} L_{en}^q + \omega_{loc} L_{loc}^q + \bar{\omega}_{loc} L_{rem}^q \\ & + \bar{\omega}_{loc} L_{tr}^q + \bar{\omega}_{loc} L_{HO}^q + L_{coop}^q, \end{aligned} \tag{4.1}$$

where L_{fg}, L_{vol}, L_{ext}, L_{ren}, L_{fc}, L_{en}, L_{loc}, L_{rem}, L_{tr}, L_{HO}, and L_{coop} are the latency due to frame generation, volumetric data generation, external sensor information generation, frame rendering, frame conversion, frame encoding, local inference, remote inference, transmission, handoff (HO), and XR cooperation. The decision of local inference is denoted as ω_{loc} and $\bar{\omega}_{loc}$ means the remote inference task, where $\omega_{loc} = \{0, 1\}$ is a binary value. Depending on the application, some segments' latency may not need to be incorporated in this model. For example, XR cooperation may be executed in parallel with rendering; therefore, it might be excluded from this calculation.

Frame generation: The delay in frame generation depends on the frame rate, frame size, allocated computation resource, data size, and the memory bandwidth of the device, which are denoted as n_{fps}, s_{f1}, c_{client}, δ_{f1}, and m_{client}, respectively. The frame size describes the complexity of a task to be processed by the XR device's computation resource. The ratio of δ_{f1} and m_{client} presents the delay in reading and writing the frame by the memory of the device. Then, the frame generation latency of the q-th frame is

$$L_{fg}^q = \frac{1}{n_{fps}^q} + \frac{s_{f1}^q}{c_{client}^q} + \frac{\delta_{f1}^q}{m_{client}^q}. \tag{4.2}$$

The values of n_{fps} (frames/s) and s_{f1} (pixel^2) are predefined in the application, δ_{f1} (MB) depends on s_{f1}, and m_{client} (GB/s) is a device configuration parameter.

Computation resource availability: The XR application requests the OS of a device (e.g., XR device or edge server) to allocate resources of certain processing units at a specific utilization ratio. The OS then allocates the resource, which cannot be expressed in an explicit analytical form easily. Using the collected experimental data, the allocated computation resource is expressed using multiple linear regression as

$$\begin{aligned} c_{client} =& \omega_c(18.24 + 1.84 f_c^2 - 6.02 f_c) \\ &+ (1-\omega_c)(193.67 + 400.96 f_g^2 - 558.29 f_g), \end{aligned} \tag{4.3}$$

where the processing speeds (i.e., the clock frequency) of CPU and GPU (GHz) are represented as f_c and f_g, respectively. The utilization rate of the CPU is denoted as ω_c, where $\omega_c \in [0, 1]$. The GPU utilization rate is $(1 - \omega_c)$, which means for a task shared by both CPU and GPU, the total resource utilization rate will be equal to 1. The R^2-value of this model is 0.87, which shows that the regression model is a good fit for the data. Note that this equation can also accommodate the allocation of TPU or NPUs depending on the data availability for proper training of the regression model.

Volumetric data generation: The latency due to volumetric data generation depends on the computation resource availability, data size, memory bandwidth, and the virtual scene size. Therefore, the latency for volumetric data generation for the q-th frame becomes

$$L_{vol}^q = \frac{s_{vol}^q}{c_{client}^q} + \frac{\delta_{vol}^q}{m_{client}^q}, \tag{4.4}$$

where s_{vol} and δ_{vol} are the virtual scene size (pixel^2) and the corresponding data size

(MB).

External sensor information generation: Assume the external sensors and devices are $m \in \{0, 1, ..., M\}$, and the XR application requires $n \in \{0, 1, ..., N\}$, updates during one frame processing time. Denote the m-th sensor's latency at the n-th update as L_{ext}^{mn}. Then the total latency for external sensor information generation for frame q becomes,

$$L_{ext}^{q} = \max_{m=0}^{M} \sum_{n=1}^{N} L_{ext}^{mnq}. \tag{4.5}$$

L_{ext}^{mnq} depends on the information generation frequency of the m-th sensor and the propagation delay between the sensor and the XR device. Denote the distance between the m-th sensor and the XR device at the n-th update during q-th frame processing time as d^{mnq} (m). Then L_{ext}^{mnq} becomes

$$L_{ext}^{mnq} = \frac{1}{f_t^m} + \frac{d^{mnq}}{c}, \tag{4.6}$$

where f_t^m and c are the information generation frequency by the m-th sensor (Hz) and the propagation speed (m/s), respectively. It is assumed that there are no path loss, shadowing, or fading effects in this propagation, which can be incorporated into the model according to system requirements.

Frame rendering: Frame rendering delay consists of the latency due to computation and file reading and writing. Three types of data are queued in the input buffer: captured frame, volumetric data, and external information, where the associated buffering delays are denoted as t_f^{buff}, t_{vol}^{buff}, and t_{ext}^{buff}, respectively. Then the delay due to data buffering during the q-th frame processing time becomes

$$t_{buff}^{q} = t_f^{buff} + t_{vol}^{buff} + t_{ext}^{buff}. \tag{4.7}$$

Assume data buffering in the input buffer is modeled as a stable $M/M/1$ queueing system. Then the buffering time of these data in the system would be $\frac{1}{\mu-\lambda}$, where μ is the average service rate and λ is the average arrival rate to the buffer with a Poisson distribution.

Additionally, the frame conversion or encoding and the local or remote inference take place in parallel to the rendering; hence, they are ignored in the calculation of the rendering delay, which contributes to the end-to-end latency, L_{tot}. Therefore, the total rendering latency can be expressed as

$$\begin{aligned} L^q_{renTotal} =& \frac{s^q_{f1}}{c^q_{client}} + \frac{\delta^q_{f1}}{m^q_{client}} + t^q_{buff} \\ &+ \omega_{loc} L^q_{tr(loc)} + \bar{\omega}_{loc} L^q_{tr(rem)}, \end{aligned} \tag{4.8}$$

where $L_{tr(loc)}$ and $L_{tr(rem)}$ are the latency for transmission of local and remote inference results to the renderer, respectively.

Frame conversion: The latency of color conversion, scaling, and cropping of frames depends on the computational resources of the device. Therefore, the frame conversion latency during frame q's processing time can be written as

$$L^q_{fc} = \frac{s^q_{f1}}{c^q_{client}} + \frac{\delta^q_{f1}}{m^q_{client}}. \tag{4.9}$$

Frame encoding: As mentioned earlier, frame encoding delay is dependent on the encoding scheme's different parameters, such as the intervals of I-frame and B-frame, bitrate, frame size, frame rate, and quantization value, which are denoted here as n_i, n_b, $n_{bitrate}$, s_{f1}, n_{fps}, and n_{quant}, respectively. However, the relation between the encoding latency and these parameters is challenging to express in an explicit analytical form. Instead, multiple linear regression is used to find the encoding latency at the q-th frame as

$$\begin{aligned} L_{en}^{q} = \Big(&-574.36 - 7.71n_i + 142.61n_b + 53.38n_{bitrate} + 1.43s_{f1}^{q} \\ &+ 163.65n_{fps} + 3.62n_{quant})/c_{client}^{q} + \frac{\delta_{f1}^{q}}{m_{client}^{q}}. \end{aligned} \tag{4.10}$$

In this model, $\delta_{f1}^{q}/m_{client}^{q}$ is included since the frames need to be read from the input buffer before encoding. The R^2-value of this regression model is 0.79, which shows a good fitness of the model.

Local inference: The local inference latency model is a function of the computation complexity of a task, allocated computation resources, data writing and reading delays, and the complexity of the CNN. Hence, the local inference computation latency for frame q can be modeled as

$$L_{loc}^{q} = \omega_{client}\left[\frac{s_{f2}^{q}}{c_{client}^{q} \cdot C_{CNN(loc)}} + \frac{\delta_{f2}^{q}}{m_{client}^{q}}\right], \tag{4.11}$$

where s_{f2} and δ_{f2} are the converted frame size and data size, respectively. Due to the activity of the image reader, $\delta_{f2}^{q}/m_{client}^{q}$ is added in this model. The complexity of the lightweight CNN stored on the XR device is denoted by $C_{CNN(loc)}$. Another variable ω_{client} determines the portion of the split task to the XR device, where $\omega_{client} \in [0, 1]$.

CNN model complexity in XR performance analysis: A CNN model's complexity depends on two parameters: the depth (the number of layers) and the size (occupied storage space on the device memory) of the CNN [20, 22]. Generally, an XR application uses a pre-trained CNN model. Therefore, the complexity of a CNN model is only considered for the inference tasks, not the training. *Note that this CNN complexity is exclusive to XR applications' latency and energy analysis. It is not applicable to other applications.* Furthermore, efficient CNN models are equipped with depth scaling nowadays [135], which is also addressed in this research. Using linear regression, the complexity of a CNN model is found as

$$C_{CNN} = 2.45 + 0.0025d_{CNN} + 0.03s_{CNN} + 0.0029d_{scale}, \tag{4.12}$$

where d_{CNN}, s_{CNN}, and d_{scale} are the depth, size, and depth scaling factor of a CNN model, respectively. The R^2-value of this model is 0.844. The regression model is trained with a vast dataset of different CNN models' latency and energy consumption data, which is later discussed in Section 4.5.

Remote inference: Remote inference in an XR application pipeline takes care of two tasks: decoding the received frame and running inference on the decoded frame. Therefore, the latency due to remote inference on a single edge server for the q-th frame can be modeled as

$$L^q_{rem} = \omega_{edge}\left[\frac{s^q_{f3}}{c^q_\epsilon \cdot C_{CNN(rem)}} + \frac{\delta^q_{f3}}{m^q_\epsilon} + L^q_{dec}\right], \tag{4.13}$$

where s_{f3} and δ_{f3} are encoded frame size and data size, c_ϵ and m_ϵ are allocated computational resources and memory bandwidth of the edge server, $C_{CNN(rem)}$ is the complexity of the large CNN model(s) running on the edge server, and ω_{edge} represents the portion of the split task to the edge server.

Decoding is usually faster than encoding due to the straightforward reconstruction of videos, typically with the help of GPUs and dedicated hardware decoders, while encoding has to deal with many issues, such as removing redundancies and predicting frames. Through the experiments, the decoding delay is found to be around one-third of the encoding delay if conducted on the same device. The percentage of the encoding delay equal to the decoding delay is defined as the discount rate, γ, in this research. Therefore, the decoding delay for the same frame can be expressed as

$$L_{dec} = \frac{L_{en} \cdot c_{client} \cdot \gamma}{c_\epsilon}. \tag{4.14}$$

Using the experimental data, a relation between the computational resources of the XR device and edge server is derived from this equation as $c_\epsilon = 11.76 c_{client}$.

Remote inference on multiple edge servers: An XR application can split the inference task on multiple edge servers that execute the task in parallel. Let E be the set of edge servers Hence, the latency due to remote inference on multiple edge servers at frame q becomes

$$L_{rem}^{q} = \max_{e=1}^{E} \left[\omega_{edge}^{e} L_{rem}^{eq} \right], \tag{4.15}$$

where, ω_{edge}^{e} denotes part of the inference task assigned to the e-th edge server. Here, $\omega_{client} + \sum_{e=1}^{E} \omega_{edge}^{e} = \omega_{task}$ where ω_{task} expresses the total inference task required by the XR application for a single frame. Each edge server's latency, L_{rem}^{eq} at frame q can be modeled according to (4.13).

Transmission latency: In the XR application pipeline, all data coming from and to the edge server are transmitted via a wireless medium. The transmission latency, therefore, is a function of the data rate and propagation delay, which at frame q can be expressed as

$$L_{tr}^{q} = \frac{\delta_{f3}^{q}}{r_w^{q}} + \frac{d_\epsilon^{q}}{c}, \tag{4.16}$$

where r_w and d_ϵ are available wireless resources (network throughput in Mbps) and the distance between the edge server and the XR device (m). Wireless path loss is not considered in this model but can be added to the framework based on system requirements.

Latency due to handoff (HO): In the proposed framework, an XR device is considered leaving a wireless coverage zone toward another with the same or different access technology, with the mobility of the XR device modeled by the Random walk model. The probability that an XR device moves from one wireless coverage zone to another, $P(HO)$, can be derived using methods in existing papers such as [136]. In addition, a vertical HO is considered for XR applications to find the HO latency, l_{HO}

using a similar analysis as in [137, 138]. The average total HO latency during frame q's processing time is

$$L_{HO}^{q} = l_{HO} \cdot P(HO). \tag{4.17}$$

XR cooperation latency: XR cooperation takes place between the client XR device and other cooperative devices via a wireless medium. Generally, this segment is executed in parallel with rendering, which is why the latency due to XR cooperation does not need to be considered in the end-to-end latency of an XR application. However, this parallel operation is entirely dependent upon the application's scope. Application developers or researchers can still use the following model to separately evaluate an XR application's performance during cooperation or include this latency into the overall latency calculation if needed. The latency due to XR cooperation is expressed as

$$L_{coop}^{q} = \frac{\delta_{f4}^{q}}{r_{w}^{q}} + \frac{d_{coop}^{q}}{c}, \tag{4.18}$$

where δ_{f4} and d_{coop} represent the data size to be transmitted to the cooperative XR device and the distance between the two communicating devices.

4.3 The Energy Consumption Analysis Model

4.3.1 Challenges in Energy Modeling

The energy consumption analysis model for XR applications is not as straightforward as measuring the power consumption only. Before proposing an energy model, there has to be sufficient insight into the power consumption behavior of an XR device during the application. For example, the power consumption trend is not the same for all the XR devices running the same application. To tackle this challenge, an analysis of a huge experimental power consumption dataset is required.

Additionally, the energy from the battery of an XR device is not entirely used for the XR application. Electrical energy is usually converted to thermal energy by a small percentage. Moreover, no matter whether the XR application is running or not, there is always a small amount of power consumption on the XR device, known as base energy. Therefore, proposing an accurate energy model for XR applications is a challenging task.

4.3.2 The Proposed Energy Consumption Analysis Model

The proposed energy consumption analysis model for an XR device while running an XR application follows a similar procedure to the latency model described in Section 4.2. The total energy consumption by an XR device for the q-th frame can be expressed as

$$\begin{aligned} E_{tot}^{q} &= E_{fg}^{q} + E_{vol}^{q} + E_{ext}^{q} + E_{ren}^{q} + \omega_{loc} E_{fc}^{q} \\ &\quad + \bar{\omega}_{loc} E_{en}^{q} + \omega_{loc} E_{loc}^{q} + \bar{\omega}_{loc} E_{rem}^{q} \\ &\quad + \bar{\omega}_{loc} E_{tr}^{q} + \bar{\omega}_{loc} E_{HO}^{q} + E_{coop}^{q} + E_{\theta}^{q} + E_{base}^{q}, \end{aligned} \tag{4.19}$$

where E_{fg}, E_{vol}, E_{ext}, E_{ren}, E_{fc}, E_{en}, E_{loc}, E_{rem}, E_{tr}, E_{HO}, E_{coop}, E_{θ}, and E_{base} represents energy consumption by the XR device during frame generation, volumetric data generation, external information generation, frame rendering, frame conversion, frame encoding, local inference, remote inference, transmission, HO, cooperation, energy converted to thermal energy, and base energy, respectively. Here, (4.19) can be further expressed as

$$\begin{aligned} E_{tot}^{q} &= \Bigg(\int_{0}^{L_{fg}^{q}} P_{fg}^{q}\, dt + \int_{0}^{L_{vol}^{q}} P_{vol}^{q}\, dt + \int_{0}^{L_{ext}^{q}} P_{ext}^{q}\, dt \\ &\quad + \int_{0}^{L_{ren}} P_{ren}^{q}\, dt + \omega_{loc} \int_{0}^{L_{fc}^{q}} P_{fc}^{q}\, dt + \bar{\omega}_{loc} \int_{0}^{L_{en}^{q}} P_{en}^{q}\, dt \\ &\quad + \omega_{loc} \int_{0}^{L_{loc}^{q}} P_{loc}^{q}\, dt + \bar{\omega}_{loc} \int_{0}^{L_{rem}^{q}} P_{rem}^{q}\, dt \\ &\quad + \bar{\omega}_{loc} \int_{0}^{L_{tr}^{q}} P_{tr}^{q}\, dt + \int_{0}^{L_{HO}^{q}} P_{HO}^{q}\, dt + \int_{0}^{L_{coop}^{q}} P_{coop}^{q}\, dt \\ &\quad + E_{\theta}^{q} + E_{base}^{q}, \end{aligned} \tag{4.20}$$

where P_{fg}, P_{vol}, P_{ext}, P_{ren}, P_{fc}, P_{en}, P_{loc}, P_{rem}, P_{tr}, P_{HO}, and P_{coop} represents power consumed by the XR device during frame generation, volumetric data generation, external information generation, frame rendering, frame conversion, frame encoding, local inference, remote inference, transmission, HO, and XR cooperation. It is found that power consumption is a function of the computational resources allocated for the task. The relationships between power and computational resources, base power, and heat dissipation in an XR device are discussed below.

Computing resource dependent power consumption: Power consumption during an XR application in an XR device is a function of the computational resources allocated for the task. However, there has been no mathematical formula to establish this relationship. As a result, the mean power consumption model of an XR application is found based on multiple linear regressions using the collected experimental dataset as

$$\begin{aligned} P_{mean} = \omega_c(18.85 f_c - 3.64 f_c^2 - 20.74) + \\ (1-\omega_c)(187.48 f_g - 135.11 f_g^2 - 62.197). \end{aligned} \tag{4.21}$$

P_{mean} is a function of both CPU and GPU resource utilization. The R^2-value of this model in (4.21) is 0.863. Interested researchers can extend this relation to accommodate TPU and NPU resources as well.

Base power in an XR device: The base power is defined as the small portion of the total power consumption that is always consumed during an XR application due to the minimal background activities and leakage current in the device. The minimal background activities are run by the OS itself, such as the system clock, display, and connectivity functions. Moreover, the leakage current is a semiconductor property that flows even if the XR device is idle. These background activities and the leakage current gives rise to a small amount of energy consumption throughout an XR application, which is denoted as E_{base} in this research.

Heat dissipation: Heat dissipation in an XR device causes serious discomfort among the users. This heat is generated by the CPU, GPU, and battery of an XR device. A small portion of the total energy consumption is converted into thermal energy and consequently dissipates heat, which is represented by E_θ in (4.19) and (4.20).

4.4 The Age-of-Information (AoI) Analysis Model

4.4.1 Challenges in AoI Modeling

In the XR application scenario presented in Fig. 4.1, the XR device is connected to a number of external sensors and devices that provide the XR device with control and environmental information, as well as scene and other data for cooperative XR. These sensors and devices are *heterogeneous* in characteristics, i.e., different information generation frequencies, different arrival rates to the input buffer, and different service rates by the buffer for each of the information packets. This makes mathematical modeling of AoI difficult.

4.4.2 The Proposed AoI Analysis Model

In this research, a sample scenario is provided in Fig. 4.2, where three sensors are generating and transmitting information. At time $t = 0$, the sensors start generating the information. However, the information generation by all the sensors is not finished at the same time, $t = 1$. This can happen due to different information generation frequencies of different sensors and devices. This frequency of the m-th sensor or device is denoted by f_t^m, where $m \in M$ and M is the set of external sensors and devices connected to the XR device. Note that the packet length is the same for all the sensors in this scenario.

Average service time and AoI: The information packets arrive at the input buffer with different arrival rates, λ, due to differences in f_t. If the service rate at the buffer is μ, considering an $M/M/1$ stable system, the average time spent in the

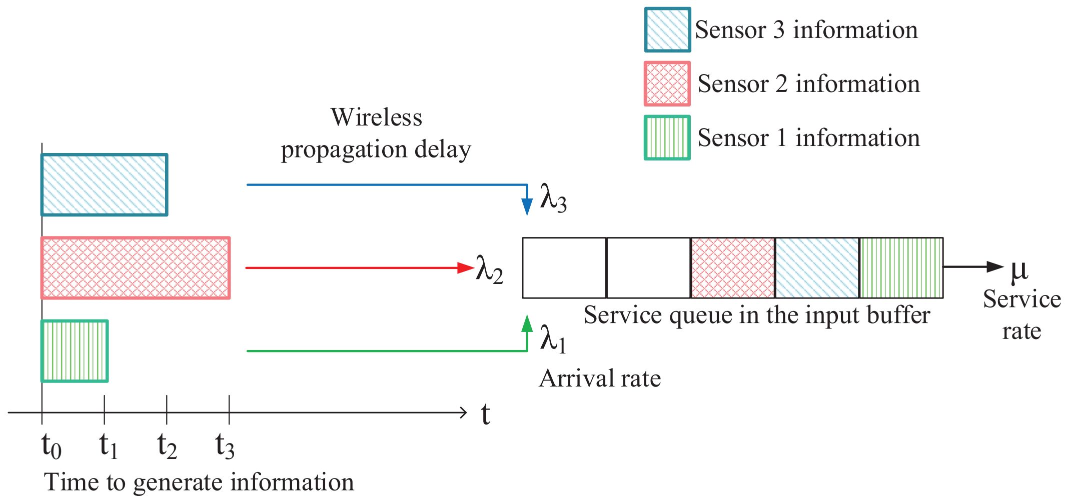


Figure 4.2: External sensor information generation, transmission, and service process for XR.

buffer by each information packet is

$$\bar{T} = \frac{1}{\mu - \lambda}. \tag{4.22}$$

The information packets experience propagation delay through the wireless medium, as well. Therefore, the AoI (in seconds) for the m-th sensor at the q-th frame is

$$t^{mnq} = T^{mnq} + \left(\frac{d_m^q}{c} + \bar{T}\right) - T_{Req}^{nq}, \tag{4.23}$$

where d_m is the distance between the m-th sensor and the XR device, c is the propagation speed, T^{mn} is the generation time of the information originated by the sensor for the n-th cycle of information generation, and T_{Req}^{n} is the time requested by the XR device to generate information for the n-th cycle. This research does not consider the wireless resources (i.e., bandwidth and TCP throughput) in this case since the control information packets are very small. Now, the average AoI of the m-th sensor at frame q's processing time is

$$A^{mq} = \frac{1}{N}\sum_{n=1}^{N} t^{mnq}, \tag{4.24}$$

Relevance-of-Information (RoI): This research introduces a new metric for XR applications in this section, which is Relevance-of-Information (RoI). RoI (no unit) is defined as the ratio of information generation frequency by an external sensor or device to the information generation frequency required by an XR application to avoid having out-of-date information in a frame. If RoI$\geq$ 1, then the information can be considered as fresh. The frequency of information processed by the XR device from the m-th sensor is

$$\bar{f}_t^m = \frac{1}{A^{mq}}. \tag{4.25}$$

The required information generation frequency is $f_{req}^m = N/L_{tot}^q$, where N is the total number of information updates during the total processing time of frame q. Then RoI can be expressed as

$$RoI = \frac{\bar{f}_t^m}{f_{req}^m}. \tag{4.26}$$

4.5 Experimental Setup and Methodology

An XR application is implemented on the devices with diverse hardware configurations listed in Table 4.1. For energy measurement, an external tool named "Monsoon Power Monitor" is used that provides power to XR devices with a data sampling rate of once at every 0.2 ms. Due to the delicate input power terminal design in these devices, the power measurement becomes challenging. Applying a heat gun and soldering process with careful investigation of the power terminals mitigates this problem. All the experiments are carried out in similar environmental conditions to make the experiments controllable and repeatable. A simple configuration of the testbed used in this research is illustrated in Fig. 4.3.

Table 4.1: Brief specifications of the XR and edge devices used in the experiments for mobile AI performance analysis

Denotation	Model	System-on-Chip	CPU	GPU	RAM	OS	Wi-Fi	Release Date
XR 1	Huawei Mate 40 Pro	Kirin 9000 (5 nm)	Octa-core (1×3.13GHzA77 3×2.54GHzA77 4×2.05GHzA55)	Mali G78	8GB LPDDR5	Android 10	802.11 a/b/g/n/ac/ax	October, 2020
XR 2	OnePlus 8 Pro	Snapdragon 865 (7 nm)	Octa-core (1×2.84GHz 3×2.42GHz 4×1.8GHz Kryo 585)	Adreno 650	8GB LPDDR5	Android 10	802.11 a/b/g/n/ac/ax	April, 2020
XR 3	Motorola One Macro	Helio P70 (12 nm)	Octa-core (4×2.0GHzA73 4×2.0GHzA53)	Mali G72	4GB LPDDR4X	Android 9	802.11 b/g/n	October, 2019
XR 4	Xiaomi Redmi Note8	Snapdragon 665 (11nm)	Octa-core (4×2GHzGold 4×1.8GHzSilver) Kryo 260	Adreno 610	4GB LPDDR4X	Android 10	802.11 a/b/g/n/ac	August, 2020
XR 5	Google Glass Enterprise Edition 2	Snapdragon XR1	Octa-Core Kryo (2×2.52GHz 6×1.7GHz)	Adreno 615	3GB LPDDR4	Android 8.1	802.11 a/g/b /n/ac	May, 2019
XR 6	Meta Quest 2	Snapdragon	Octa-core XR2 Kryo 585	Adreno 650	6GB LPDDR5	Oculus OS	802.11 a/g/b/n/ac/ax	October, 2020
External & XR 7	Nvidia Jetson TX2	Tegra	Dual-Core NVIDIA Denver2 Quad-Core A57 MPCore	256-core NVIDIA Pascal	8GB LPDDR4	Ubuntu 18.04	–	March, 2017
Edge server	Nvidia Jetson AGX Xavier	Tegra	Octa-core ARM v8.2	512-core Volta Tensor Cores	32GB LPDDR4X	Ubuntu 18.04 LTS aarch64	–	October, 2018

This research uses 11 CNN models with distinct architectures in this research that are listed in Table 4.2. A LinkSys dual-band router (2.4 GHz and 5 GHz) is used as the wireless medium, which connects the XR devices, external sensors, and the

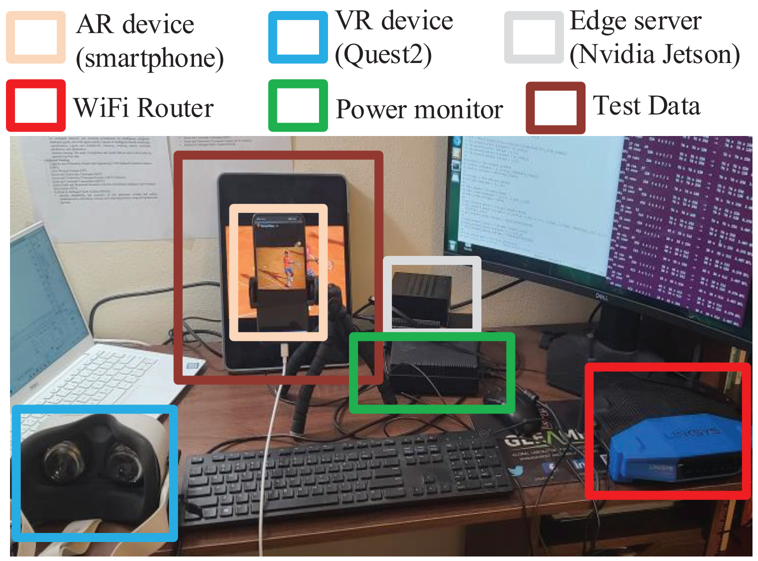


Figure 4.3: A snippet of the experimental testbed used in this research.

server via Wi-Fi. For the remote inference, `YOLOv3` and `YOLOv7` are used in the edge server. The 2017 COCO test dataset is used for testing the XR application for both local and remote inferences.

Regression model training: After conducting experiments, a huge dataset is collected for the design and evaluation of the proposed XR performance analysis modeling framework. To design and evaluate the framework, datasets containing 119, 465 and 36, 083 data, respectively, are used. The regression models are trained

Table 4.2: CNNs used in mobile AI performance analysis framework design

CNN	**Model depth (no. of layers)**	**Storage space (MB)**	**GPU support**
MobileNetv1_240 Float	31	16.9	Yes
MobileNetv1_240 Quant	31	4.3	
MobileNetv2_300 Float	99	24.2	Yes
MobileNetv2_300 Quant	112	6.9	
MobileNetv2_640 Float	155	12.3	Yes
MobileNetv2_640 Quant	167	4.5	
EfficientNet Float	62	18.6	Yes
EfficientNet Quant	65	5.4	
NasNet Float	663	21.4	Yes
YoLoV3	106	210	Yes
YoLoV7	Scaling (1.5)	142.8	Yes

with the data collected from devices XR1, XR3, XR5, and XR6, and tested with the data collected from XR2, XR4, and XR7. Training and testing with separate datasets help evaluate the models' performance. All the regression-based models used in this research are generated using a 95% confidence boundary.

4.6 Performance Evaluation

The proposed XR performance analysis model's performance is evaluated compared to the test dataset collected during experiments, which are considered "Ground Truth (GT)" in this research. First, the performance evaluation of the proposed model is presented for end-to-end latency and energy consumption of the XR pipeline. Then the AoI model's performance is evaluated based on an emulated experiment. Finally, the end-to-end latency and energy consumption analysis in different conditions by the proposed model are compared with several state-of-the-art analysis models. The proposed models' performance is evaluated for each individual segment of the XR application's pipeline considered in this research. Some of the major evaluation results are discussed in this section.

4.6.1 End-to-End Latency Validation

The end-to-end latency of the XR application calculated by the proposed analytical model is compared with the ground truth for both local (Fig. 4.4a) and remote inference (Fig. 4.4b). In remote inference, device mobility is not considered.

Insights: A mean error of 2.74% and 3.23% in local and remote inference latency calculation is observed as compared to the ground truth, which means the proposed model is very accurate in calculating the end-to-end latency for XR applications. It is observed that the more diverse the training dataset for regression is, the more accurate the model can be.

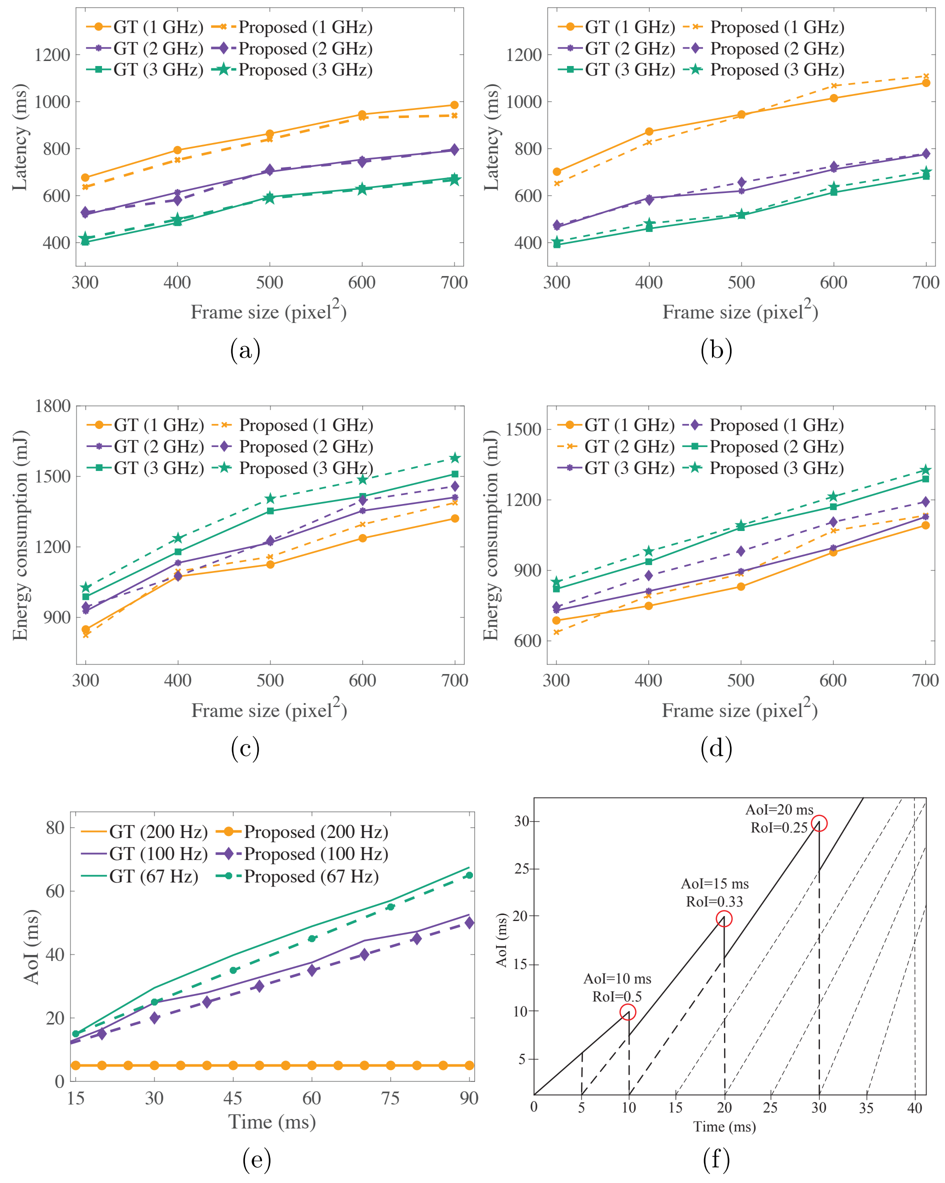


Figure 4.4: Evaluation of the proposed XR performance analysis models: analysis of end-to-end latency for (a) local and (b) remote execution, end-to-end energy consumption for (c) local and (d) remote execution, AoI at different (e) information generation frequency and (f) RoI for information generation frequency of 100 Hz.

4.6.2 End-to-End Energy Consumption Validation

The comparison of end-to-end energy consumption of the XR service obtained from the proposed model and ground truth is shown in Figs. 4.4c and 4.4d for local and remote inference, respectively.

Insights: The mean errors for local and remote inference energy calculation are 3.52% and 5.38%, respectively, as compared to the ground truth. This error percentage can be reduced by improving the regression model's performance of the power consumption model.

4.6.3 Age-of-Information Validation

A sample AoI scenario is emulated where three sensors have a rate of information generation of 1 every 5, 10, and 15 ms (200 Hz, 100 Hz, and 66.67 Hz), respectively. The XR application has a requirement of 1 information update every 5 ms. From Fig. 4.4e, an increase in AoI is observed when the rate of generating information gets lower (i.e., the larger value of frequency). The reason behind this increase is the increase in the delay between information request and information generation at every update cycle (e.g., the sensor generating information at the 67 Hz frequency is transmitting the first information when the third update is required). This incident is explained in detail in Fig. 4.4f, where the sensor with a 100 Hz information generation frequency has incremental AoI at every XR information update period. The corresponding RoI is also shown in the figure.

Insights: To maintain a proper AoI, sensors should follow the RoI and use a necessary frequency of information generation.

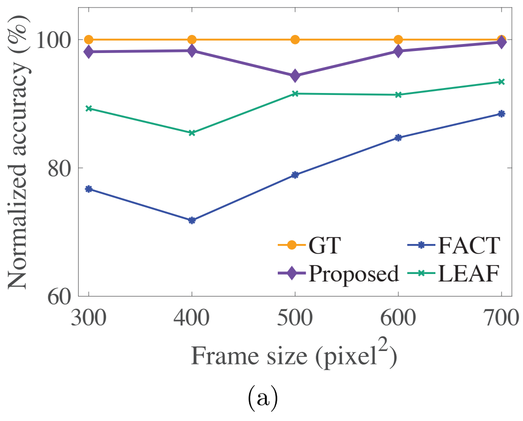
4.6.4 Comparison of Model Performance

The latency and energy of the proposed model are compared with two existing analytical models: FACT [1] and LEAF [2]. *The reason behind considering these two models is that these are the most comprehensive and accurate state-of-the-art analytical models presented by researchers so far for augmented reality applications – a subset of XR applications. No other existing models provide better insights into such applications. Therefore, these two models are the best candidates for comparison with the proposed performance analysis framework.*

- **FACT [1]:** FACT proposes to include computation, core network, and wireless latency into the overall service latency model of an edge-assisted AR application. The respective energy model can be found by following each component of the service latency. However, FACT presents the computation latency as a function of the computation complexity and available computation resources, which are formulated without considering different processing sources, data size, and the memory of the device, but they are taken into account in this research.

- **LEAF [2]:** LEAF overcomes several limitations of FACT by breaking down the entire pipeline of an edge-AR application and considering each segment's latency separately. However, it still suffers from the simplicity in formulating the computation latency and energy as FACT does. The proposed framework presents a comprehensive way to model XR latency and energy consumption to achieve more accurate performance results.

The performance comparisons for end-to-end latency and energy consumption using remote inference are shown in Figs. 4.5a and 4.5b in terms of normalized accuracy, where the normalized accuracy of GT is 100%. The proposed analytical model performs with higher accuracy than FACT and LEAF in latency calculation by 17.59%



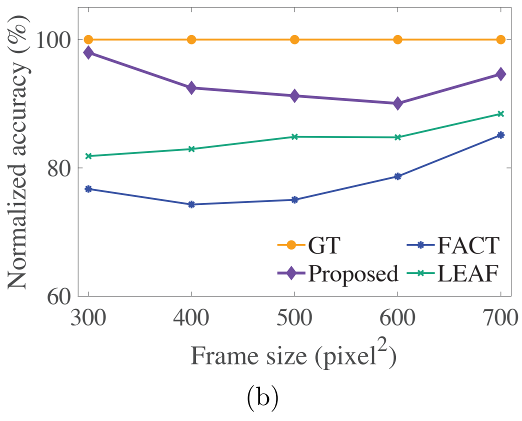


Figure 4.5: Comparison of (a) end-to-end latency and (b) end-to-end energy consumption for remote inference obtained from GT and analytical models with FACT [1] and LEAF [2] in terms of normalized accuracy.

and 7.49%, and in energy calculation by 15.30% and 8.71%, respectively.

Insights: The proposed XR performance modeling framework achieves this superior performance due to the consideration of the complex models of computation resource, encoding, and transmission, and the relation between the computation resource of the XR device and edge server.

CHAPTER 5: PROPOSED SERVICE AGGREGATION MECHANISM IN HIGH-MOBILITY AI APPLICATIONS

In this chapter, a novel service aggregation system based on periodic AoI prediction is proposed (Sec. 5.1. The AoI prediction is done using an LSTM network, which is chosen by comparing with two other machine learning models by trading off between latency, memory consumption, and accuracy. A range of period for AoI prediction is proposed that can reduce latency and memory consumption even more. The algorithm of the proposed system is also presented in this section.

Finally, the proposed system's performance is evaluated in terms of newly introduced performance metrics in Sec. 5.2. A comparison of the proposed service aggregation system with respect to three other state-of-the-art data queueing methods is also presented.

5.1 Proposed CAV Service Aggregation using Predictive AoI

In this research, it is assumed that the information update messages contain information about the source node (i.e., whether it is a stationary roadside sensor or a mobile vehicle). Moreover, all the connected vehicles exchange basic information with each other, such as speed and geolocation (e.g., latitude-longitude). Thus, the relative speed of the EGO vehicle with respect to other connected vehicles can be derived.

5.1.1 AoI Prediction

From an initial study, it is observed that there is an implicit relation between the relative speed of the ego vehicle and the AoI from specific information source nodes. The predictive AoI model for each source node, n, thus has two input parameters:

the timestamp and relative speed of E_v with respect to n. The relative speed is a function of the relative distance between E_v and n, and time.

For a low latency CAV application, the prediction of AoI should be done in such a way that it does not introduce significant additional load to the system. For instance, if the system has a certain latency requirement for each update, the prediction needs to be completed within a time frame, leaving sufficient time for service aggregation tasks to be done by the required latency. Moreover, the accuracy of prediction is of utmost importance in the case of CAVs due to the involvement of safety issues. Finally, the prediction needs to be done for each source node. With an increase in the number of sources, the computational load for prediction increases – which emphasizes the use of a low-complexity prediction model. Therefore, before choosing a prediction model, the latency, accuracy, and computing load need to be evaluated first. Three prediction models are implemented to predict the AoI in CAVs, which are linear regression, Random Forest, and long short-term memory (LSTM) network. Fig. 5.1 shows the comparison results of the three prediction models. The necessity of a trade-off is evident here since each model has different pros and cons. Being the highest priority performance metric, accuracy and latency dictate the use of the LSTM network in this research. The high accuracy provided by the LSTM network is due to its better and recurrent understanding of the temporal dependencies of the training dataset.

Though the LSTM network provides the best prediction performance in this research, it still has a high latency and memory consumption, which may not satisfy the overall QoS requirement of the CAV application (e.g., latency and computing load). Consequently, a periodic prediction system is proposed that predicts the AoI at a certain interval. Additionally, this prediction system clusters the AoI data of several sources based on similarity over a specific time period. This reduces the computational load and latency by a high margin since the prediction runs alongside

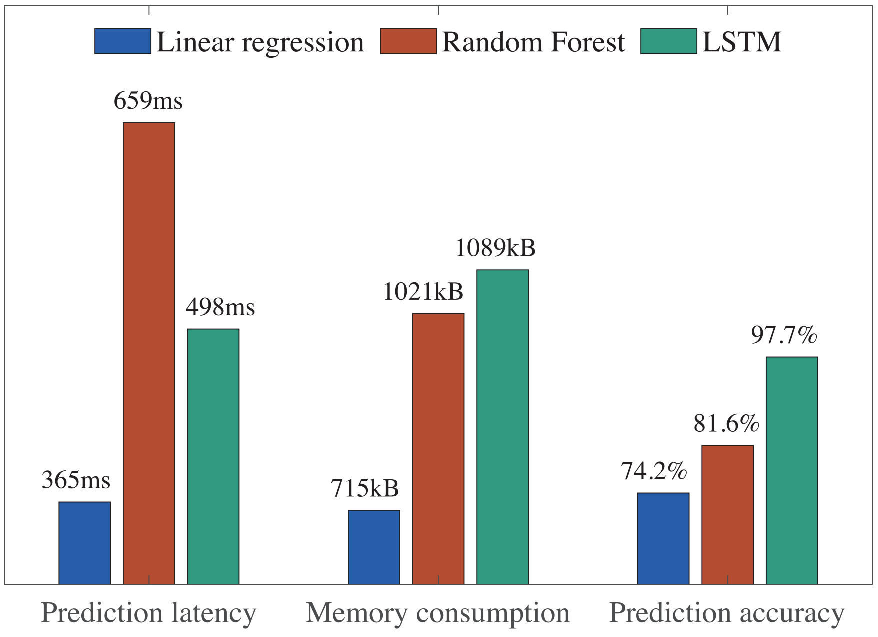


Figure 5.1: Comparison of AoI prediction latency, memory consumption, and accuracy of different prediction models.

the service aggregation tasks. Let the period of prediction be denoted as N. The predictive model predicts AoI at every N-step, and it predicts N-step-ahead AoI. N can be determined by the application developers in numerous ways that allow them to execute the prediction within the upper bound of the latency requirement. This research defines a new parameter called "speed-to-coverage area ratio (SCAR)", which is the ratio of the relative speed of E_v to the coverage area of a source node, n (source node can be either stationary or moving). The prediction should be done at least once before E_v leaves the coverage area of a source. If the prediction latency is denoted as L_{pred}, then the range of N that is recommended can be expressed as

$$N = \left[\frac{L_{pred}}{1/Q}, \frac{Q}{SCAR_n}\right], \tag{5.1}$$

where $1/Q$ is the maximum AoI threshold. Fig. 5.2 shows the performance of unit-step and a 3-step AoI prediction by an LSTM network. Using a multi-step periodic prediction, the system is able to reduce the overall latency by at least 42%. The

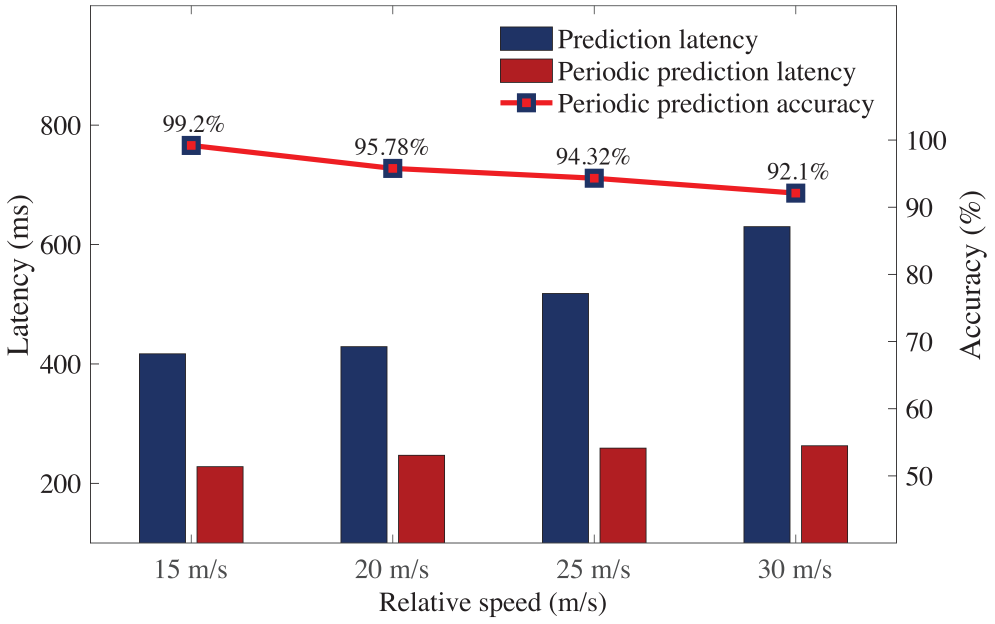


Figure 5.2: AoI prediction latency, periodic AoI prediction latency, and periodic AoI prediction accuracy using LSTM network.

hyperparameter values and types for the training of the LSTM network are shown in Table 5.1.

Table 5.1: Training Hyperparameters of LSTM Network for AoI Prediction

Hyperparameters	**Values/Type**
No. of LSTM units in each layer	64
No. of LSTM layers	4
Dropout rate	0.1
Recurrent dropout rate	0.1
Activation function	*tanh*
Weight initializer	glorot_uniform
Recurrent weight initializer	orthogonal
Training batch size	32
Training epochs	50
Optimizer	Adam
Learning rate	0.001
Loss function	mean_sqaured_error

5.1.2 Service Aggregation using Predictive AoI

The proposed service aggregation using periodic predictive AoI can be divided into five tasks. First, the system initializes two buffers: the outer one is for receiving updates from information source nodes via broadcast messaging (data buffer), and the inner one is to put updates according to their update cycle (update buffer). Second, if the system receives an update from a new node that does not belong to the node list, $S \cup V$, based on its AoI, the system determines whether to establish a new connection and put this update into the respective segment of the update buffer. Third, the system checks whether it is the period for prediction, and if it is, then the LSTM network predicts N-step-ahead AoI using the current relative speed, v, and timestamp, t. If the N-step-ahead AoI is satisfactory, then it maintains the service connection at that cycle; otherwise, it terminates the connection with the specific node. The information update is also processed accordingly. Fourth, the system clusters nodes based on the predictive AoI to reduce the computing load. Finally, the system updates the node list, $S \cup V$, and the information update set, U. This entire process is illustrated in Algorithm 1.

5.2 Performance Results and Discussion

In this section, the experimental setup, performance metrics, performance evaluation of the proposed CAV service aggregation using predictive AoI, and comparison of the proposed system with state-of-the-art methods are discussed.

5.2.1 Experimental Setup

Extensive experiments are conducted using simulated CAV scenarios with OMNet++. For the mobility model of the CAVs, a modified version of the freeway model is used from [139]. The speed of the CAVs ranges from $15 - 30$ m/s. The coverage areas of sensors and vehicles are set to be 100m and 300m, respectively. Each simulation lasts for 20 minutes. The prediction periods for sensors and vehicles are set to be

Algorithm 1 Proposed CAV service aggregation using periodic predictive AoI.

```
Input: Relative speed $v_n^m$, Timestamp $t^m$, and $AoI_n^m$.
Begin
  Initialize Hyperparameter settings, data buffer, $B$, update buffer, $U$, and period of prediction, $N_n$.
  foreach update cycle, $m$ do
      if $n$ is not in node list then
          if $AoI_n^m \leq AoI_{max}$ then
              Add $n$ to node list, $S \cup V$
              Add $U_n^m$ to $U$
          end
          else
              Discard $U_n^m$
          end
      end
      else
          foreach source node, $n$ do
              if $m \mod N_n = 0$ then
                  Input of LSTM $\leftarrow v_n^m$, $t^m$
                  Output of LSTM: $AoI_n^{m+N_n}$
                  if $AoI_n^{m+N_n} > AoI_{max}$ then
                      Terminate connection with $n$ at update cycle, $m + N_n$
                      Update node list, $S \cup V$
                  end
                  else
                      Maintain connection with $n$ at update cycle, $m + N_n$
                      Determine update segment, $U$, for $data_n$
                      Put $U_n$ to selected update segment, $U$
                  end
              end
              else
                  if $AoI_n^m \leq AoI_{max}$ then
                      Add $U_n^m$ to $U$
                      Maintain connection with $n$ at update cycle, $m$
                      Cluster $n$ with other nodes with equal $AoI^m$
                      Update node list, $S \cup V$ with cluster nodes
                  end
                  else
                      Discard $U_n^m$
                  end
              end
          end
      end
  end
return node list, $S \cup V$, and information update set, $U$
Output: Service connection decision and information update, $U_n^m$.
```

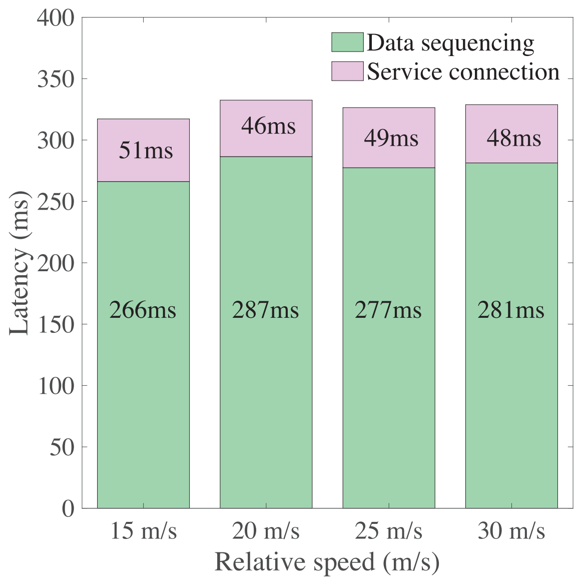
5 and 10. The training and testing dataset used for the LSTM network contains data of $96,000$ and $12,000$ timestamps, respectively. The required information update frequency is set to 3 updates per second, which makes the maximum AoI threshold 333 ms.

5.2.2 Performance Metric

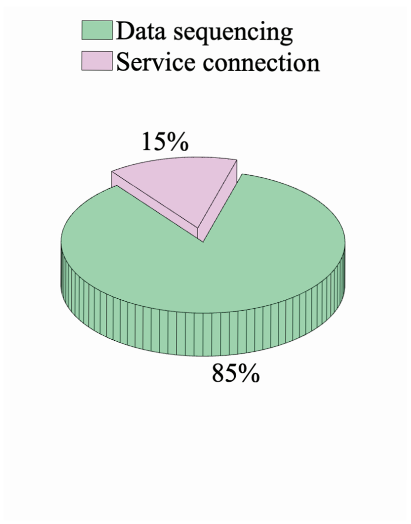
The proposed system's performance is evaluated and compared in terms of the overall latency and data sequencing success rate (DSSR) at different relative speed values of the ego vehicle and continuously varying relative speeds at different timestamps. DSSR is introduced in this research for the first time as a performance metric of CAV service aggregation, which is defined as the percentage of successful data sequencing with respect to the total number of data in a buffer segment.

5.2.3 Overall Latency of the Proposed System

The overall system latency of the proposed CAV service aggregation can be divided into two parts: data sequencing and service connection. The mean overall latency



(a)



(b)

Figure 5.3: (a) Mean system latency at different relative speed and (b) overall latency of system tasks.

at different relative speeds of the ego vehicle and the components of the system are shown in Fig. 5.3. The mean overall latency of the system is around 326 ms, whereas data sequencing takes around 85% of the total latency.

5.2.4 Proposed System's Performance at Varying Speeds

In Fig. 5.4, the proposed system's performance is shown at continuously changing relative speeds of the ego vehicle in terms of mean latency per update and DSSR. It is observed that the system is able to maintain a mean overall latency of 327 ms with a DSSR of 98% considering all the relative speeds. The initial latency (at 100 ms timestamp) of the system is a bit higher (around 337 ms) since the AoI prediction is executed at a later timestamp. Note that the highest priority of the proposed system is to maintain a latency that is under the maximum tolerable threshold (333 ms in this case), which may come at a cost of slightly reduced DSSR at a higher relative speed.

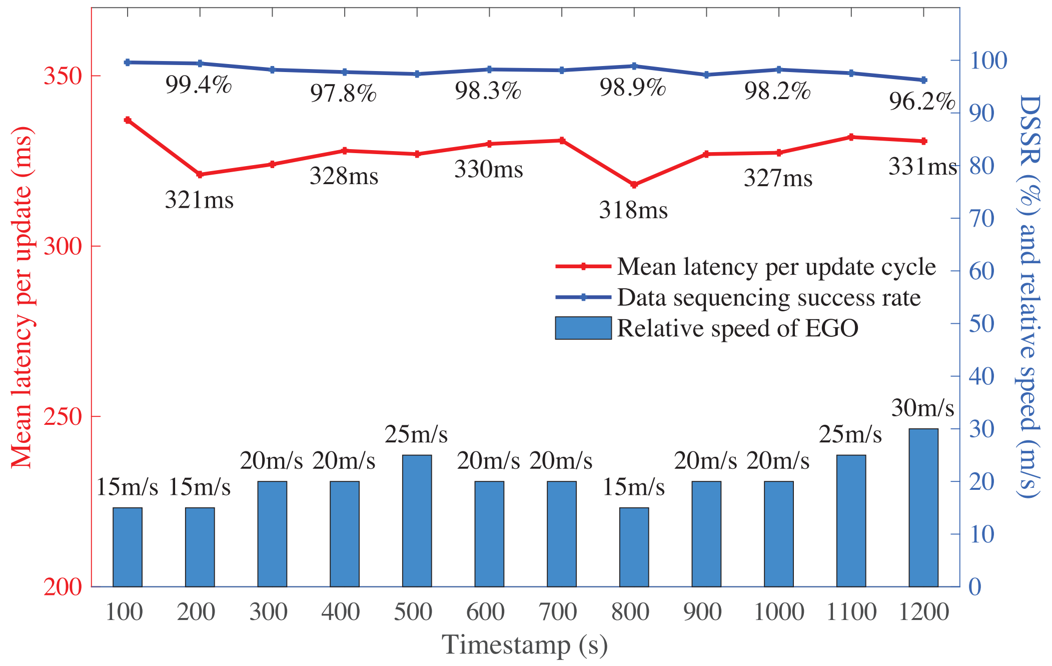


Figure 5.4: Mean system latency and data sequencing success rate (DSSR) at varying relative speed and timestamps.

5.2.5 Comparison of System DSSR and Latency

The performance of the proposed CAV service aggregation using periodic predictive AoI (denoted as "Proposed System" in this subsection) is compared with three other state-of-the-art data communication and queuing techniques listed below. Since there are no existing service aggregation methods available, these methods are modified to fit the research problem of service aggregation.

- **FIFO:** First-in-first-out (FIFO) is a common data queuing and dequeuing method, where the data are served first that arrive first.

- **Stop-N-Wait:** This is a popular data link and transport layer protocol for data communication, which is modified for this research to stop and wait for the data in the correct information update cycle.

- **Priority Queue:** This is another queuing technique that assigns different priorities to data and serves according to the pre-set priorities. In this research, information updates from nearby vehicles are given a higher priority.

The comparison among the four service aggregation methods is shown in terms of mean DSSR (%) and mean overall latency (ms) per update at different relative speeds of the ego vehicle in Fig. 5.5 and 5.6, respectively. DSSR in FIFO declines sharply with an increase in speed since the ego vehicle passes the coverage area but does not terminate the connection or adjust its data buffer accordingly. Priority queue also experiences a declining DSSR with an increase in speed due to a similar effect. An interesting finding here is that the DSSR is almost the same in the case of Stop-N-Wait and Predictive AoI in every relative speed because both methods wait for the correct information update. However, the mean latency is much higher in Stop-N-Wait because of the high waiting time. Predictive AoI can achieve 78% lower latency due to its periodic prediction and clustering.

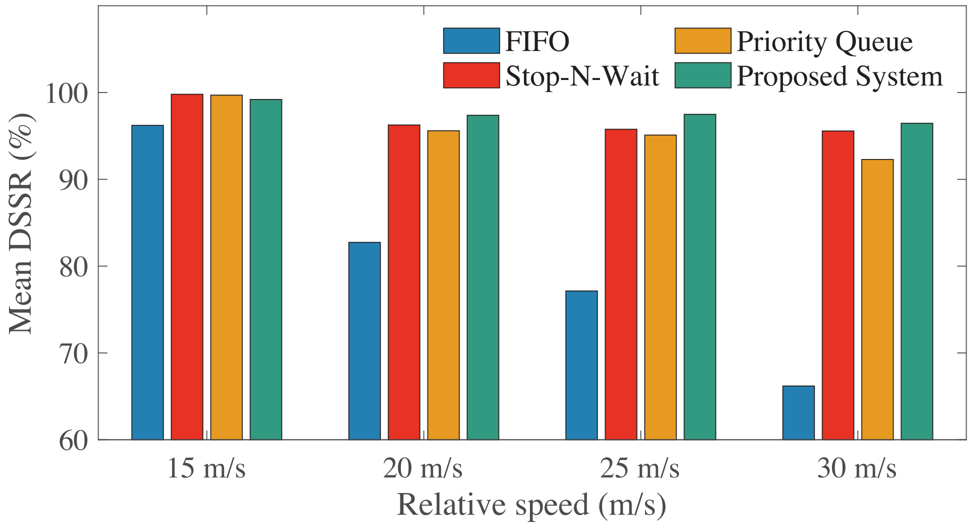


Figure 5.5: Comparison of mean data sequencing success rate per update cycle.

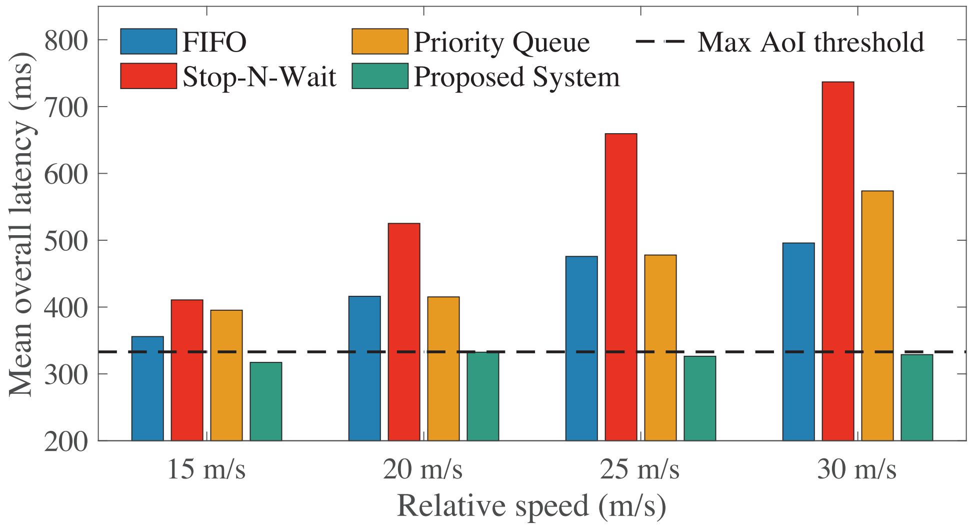


Figure 5.6: Comparison of mean overall latency per update cycle.

The average increase in DSSR is 7%, and the decrease in latency is 51% for the proposed predictive AoI-based CAV service aggregation system compared to the other three methods. Additionally, the predictive AoI-based system successfully maintains the average AoI below the maximum AoI threshold.

CHAPTER 6: PROPOSED MOBILE AI PERFORMANCE IMPROVEMENT TECHNIQUE IN UNSTABLE WIRELESS NETWORK

In this chapter, a novel H.264 video encoding-based edge-assisted MAR system is proposed to overcome the challenges due to unstable wireless network conditions. The proposed system is described in Sec 6.1. This encoding scheme is chosen for the system due to its high compression ratio and compatibility with object recognition systems. Before designing this system, experiments are carried out with different smartphones running a basic Edge-MAR application (object detection). The experimental measurements of the total latency and energy consumption show the necessity of applying compression in data transmission.

Additionally, he transmitted data size and inference accuracy are measured and collected for an Edge-MAR application with H.264 encoding and compared with those for an existing Edge-MAR application. The study shows that there is a trade-off between data size and accuracy in order to use any of the systems. Moreover, three different wireless transmission conditions are emulated in the testbed. The latency and energy data for both systems are analyzed. It is evident that only the data transmission part of an Edge-MAR pipeline gets affected due to worse wireless signal strength. However, introducing encoding in an Edge-MAR system helps reduce the latency and transmitted data size, but at the cost of additional energy consumption. These findings are presented in Sec. 6.3. Lastly, multiple linear regression-based models are proposed in this section that can predict the MAR system performance and required parameter settings with high accuracy.

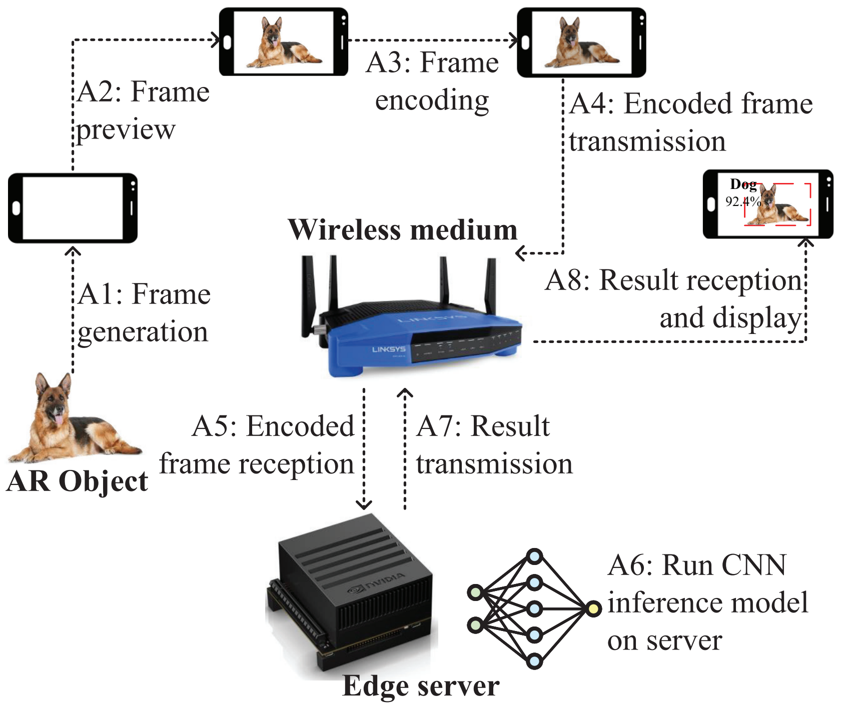

A2: Frame preview
A3: Frame encoding
A4: Encoded frame transmission
Dog 92.4%
Wireless medium
A1: Frame generation
A8: Result reception and display
AR Object
A5: Encoded frame reception
A7: Result transmission
A6: Run CNN inference model on server
Edge server

Figure 6.1: System description of Edge-MAR with H.264.

6.1 Proposed System Description

This section presents an Edge-MAR system based on the H.264 video encoding scheme to detect and recognize objects. This system includes an encoder on the client side (mobile devices) and a decoder on the server side. Like other Edge-MAR pipelines, the proposed system does not include a “frame conversion” segment since the encoder can process raw frames with YUV color formats. The system workflow is shown in Fig. 6.1.

A1: A frame is generated by the client’s camera capturing the intended AR object with the available background.

A2: The raw frame is previewed on the client’s output display.

A3: The raw frame is sent to the encoder of the client and further encoded using the H.264 scheme.

A4: The encoded frame is transmitted to the edge server over the wireless network. This communication takes place by creating a TCP (transmission control protocol) socket at the server end.

A5: The server receives the encoded frame, and then decodes it. If running CNN on each frame takes longer than the frame reception time, all the received and decoded frames remain in the queue (buffer).

A6: The server runs the CNN inference model on each decoded frame and gets the result of the object detection and recognition.

A7: The result is then transmitted back to the client using the TCP socket.

A8: The client receives the result for each frame and displays it with a bounding box and inference accuracy.

This workflow is repeated while the MAR application is running.

6.2 Experimental Setup and Methodology

6.2.1 Testbed

Vast experiments are performed using the testbed implemented for this research, consisting of different smartphones (as mobile client devices), an edge server, and a WiFi router. In order to make the proposed system usable for most of the commercially available Android OS-based smartphones, phones released in different years, having different specifications are selected, which are listed in Table 6.1. A Jetson AGX Xavier [140] is used as the edge server, which has an 8-core ARM 64-bit CPU with 32GB 256-bit LPDDR4x 137GB/s RAM and 512-core Volta GPU with Tensor Cores. As the WiFi access point, a Linksys dual-band router [141] is used, which is connected to the edge server. For energy consumption measurement purposes, an external instrument – "Monsoon Power Monitor" [142] is connected to smartphones.

6.2.2 Methodology

For "Edge-MAR with H.264", Android's "MediaCodec" library is used to encode the generated frames from the mobile device's camera. The camera is moved at a constant speed and angle to capture the objects to be detected. MediaCodec pro-

Table 6.1: Brief specifications of the smartphones used in the experiments of mobile AI performance improvement in unstable wireless network

Phones	**Samsung**	**Asus**	**Motorola**	**Vivo**	**Google**
Model	Galaxy S5	ZenFone	One Macro AR	IQOO Z1 (XT2016-2)	Pixel 4a
OS	Android 6.0.1	Android 7.0	Android 9.0	Android 10.0	Android 10.0
SoC	Snapdragon 801 (28nm)	Snapdragon 821 (14nm)	MediaTek (12nm)	Mediatek (7nm)	Snapdragon 730G (8nm)
CPU	32-bit 4-core 2.5GHz Krait 400	64-bit 4-core 2.4GHz Kryo	64-bit 8-core 2×2GHz A73 2×2GHz A53	8-core (4×2.6GHz & 4×2GHz Cortex)	8-core (2×2.2GHz & 6×1.8GHz Kryo)
GPU	Adreno 330	Adreno 530	Mali-G72	Mali-G77	Adreno 618
RAM	2GB	6GB	4GB	6GB	6GB
WiFi	802.11 n/ac	802.11 n/ac/ad	802.11 b/g/n	802.11 a/b/g/n/ac	802.11 a/b/g/n/ac
Release date	April, 2014	July, 2017	October, 2019	May, 2020	August, 2020

vides a compression ratio of around 92% (1:12.5) in this experiment. For ease of development, 300 frames are saved into a video file. Then the file is encoded and transmitted to the edge server. The encoding parameters used in this experiment are 30 fps, I-frame interval 5, and a maximum video bitrate of 30 MB/s. For remote execution in the edge server, a version of the famous CNN model, YOLOv3 [143], is adopted that uses the COCO dataset [144] having 80 classes of objects. The 2.4 GHz band of the router is used to access the Wi-Fi network. To produce different network conditions with different RSS, different distances are emulated with line-of-sight considerations from the Wi-Fi access point to the mobile devices while keeping the transceivers directional.

The energy consumption of smartphones is measured by the power monitor that is connected to the smartphones via the battery terminals. In the case of the latest smartphones, the batteries need to be removed from the back panel of the phones by applying heat from a heat gun. Then the terminals are soldered to extended wires, which are then connected to the input/output terminals of the power monitor — the power monitor powers up the phones.

This external power monitor provides voltage, current, and energy consumption data every 2 ms. Before measuring the energy consumption, all the irrelevant features and background applications are turned off in the smartphones to understand the behavior of the object detection application properly. The latency data for different segments of the Edge-MAR pipeline, on the other hand, are logged in separate files. The application is run for 300 frames each time. Then the energy and latency are considered for a single frame by taking the average of all the measurements for 300 frames. To compare the experimental results with an existing MAR system, the work in [7] is implemented, which is named here as "Edge-MAR" only. Finally, using multiple linear regression, new models are developed for different parameters of the Edge-MAR system taking all the experimental data as inputs.

6.3 Performance Analysis of Edge-Assisted MAR Systems

6.3.1 Key parameters

6.3.1.1 Performance metrics

In any edge-based AR system, latency is the most important performance metric, which defines whether a system is suited for real-time or other sensitive applications. The inference accuracy describes the system's ability to recognize any object correctly. Moreover, for Edge-MAR systems, energy consumption is another crucial metric to determine a mobile device's stability in terms of battery health. Lastly, the transmitted data size dictates how much network resources are consumed.

6.3.1.2 Control factors

The experimental testbed consists of an H.264 encoder, where the encoding configuration regulates the encoding latency and energy consumption due to encoding. Additionally, smartphones' CPU frequency governs the way frames are processed and encoded. The size of the captured frames does not necessarily control the compression, but the data size depends on it heavily. However, no matter what the data size is, the transmission latency and energy vary on different signal strengths of the wireless medium.

6.3.2 Latency and energy consumption of Edge-MAR and Edge-MAR with H.264 encoding

Experiments are conducted on both Edge-MAR and Edge-MAR with H.264 for 8 different sizes of frame resolution (300×300, 350×350, 400×400, 450×450, 500×500, 600×600, 700×700, and 800×800) and for 3 different CPU frequencies (1, 2, and 3 GHz). The main difference between the pipelines of these two systems is the presence and absence of frame conversion and frame encoding, and vice-versa. The latency and energy measurements are shown in Fig. 6.2.

The overall latency and energy consumption for Edge-MAR varies from 677.6 ms to 1156.35 ms and 5.87 J to 7.55 J for 1 GHz, 662.84 ms to 1144 ms and $6.45J$ to 7.81 J for 2 GHz, and 610.96 ms to 1115.5 ms and 6.58 J to 8.6 J for 3 GHz CPU frequency for the above-mentioned frame sizes. The major latency is caused by the transmission, and most of the energy is consumed by the frame generation. It is observed that *for frame sizes from* 350×350*, the latency does not increase drastically for Edge-MAR till the frame size of* 500×500*. After that, the change in latency is steeper.* Similar trend goes for *energy consumption* also in Edge-MAR.

For Edge-MAR with H.264, it is found that *the overall latency is reduced by around* 80%*, but at the cost of energy consumption increase of around* 20%*. Most of the* *latency and energy consumption is here caused by the encoding.* Similar to Edge-MAR, in this system, from frame size 350 × 350 to 500 × 500, latency and energy consumption do not increase significantly.

Insight: With the increase in CPU frequency, latency decreases, and energy consumption increases. However, for the increase in frame resolution, both latency and energy consumption rise. Edge-MAR with H.264 provides less latency but at the cost of an apparent increase in energy consumption. Frame size 350 × 350 to 500 × 500 is observed to be an optimal range for MAR applications in terms of latency and energy consumption due to the hardware limitations such as sensor size and frame rates of mobile devices.

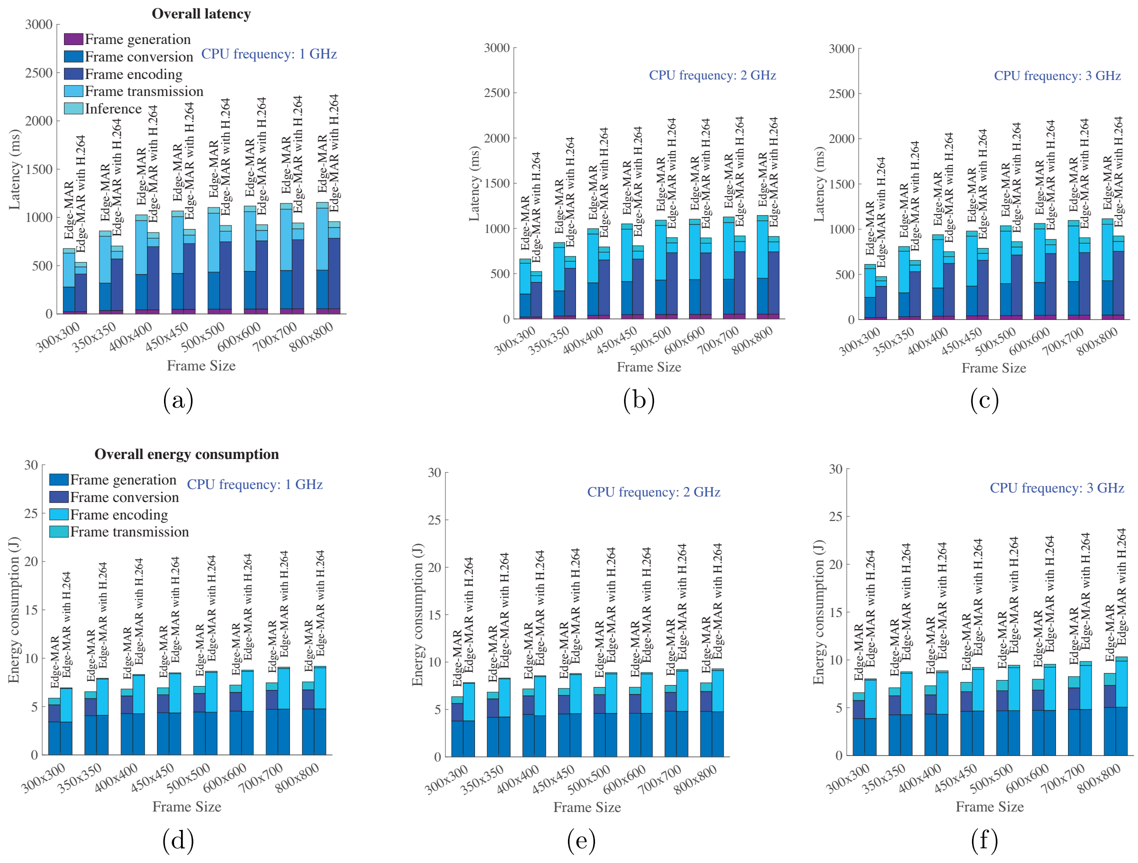


Figure 6.2: Overall latency at CPU frequency (a) 1 GHz, (b) 2 GHz, and (c) 3 GHz, and energy consumption at CPU frequency (d) 1 GHz, (e) 2 GHz, and (f) 3 GHz for Edge-MAR and Edge-MAR with H.264 respectively.

6.3.3 Data size and accuracy

The measured data show that with the increase in frame sizes, the size of transmitted data per frame rises, as shown in Fig. 6.3. This is due to the increase in frame information with the larger frame size. From frame size 400 × 400 to 450 × 450 and from 600 × 600 to 700 × 700, there is a sharp rise in data size due to the sudden introduction of additional information. *The increment in data size is more stable from frame sizes* 450 × 450 *to* 500 × 500 *because of a more minor increase in background*

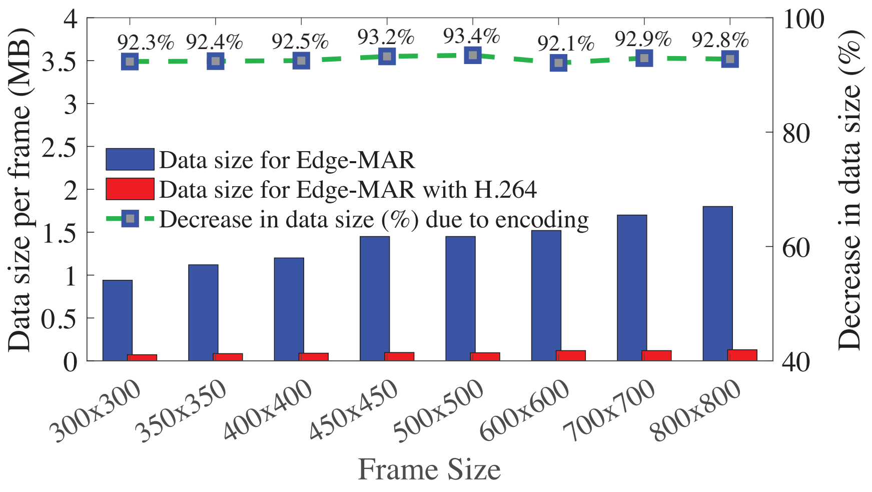


Figure 6.3: Transmitted data size and decrease due to encoding for different frame sizes and CPU frequencies.

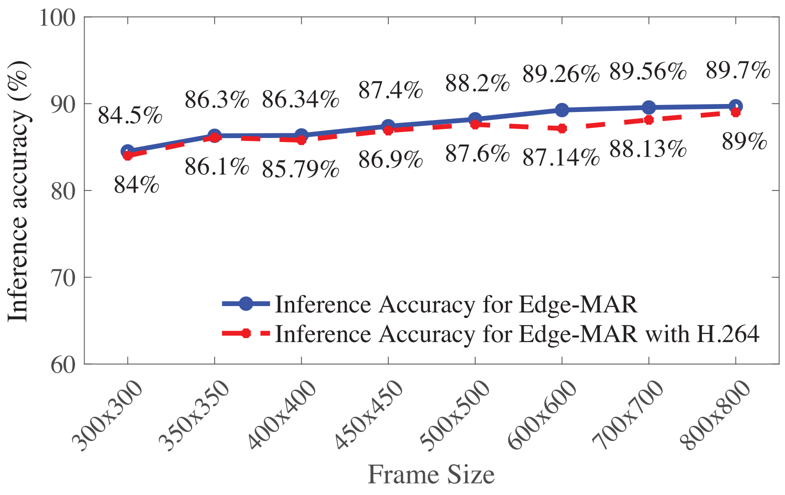


Figure 6.4: Inference accuracy in percentage for different frame sizes and CPU frequencies.

information with the movement of the camera or the object. This is true for the encoded data size as well. Another interesting finding is that *there is a slight decline in compression at frame sizes* 600×600 *and* 800×800 *because of small information added to the frames compared to the other sizes.* The data size for Edge-MAR varies from 0.95 MB to 1.8 MB per frame, and for Edge-MAR with H.264, from 0.072 MB to 0.13 MB per frame with an increase in frame sizes, i.e., additional information. Furthermore, the inference accuracy varies a little across different frame sizes. It ranges from 84.50% to 89.7% per frame, with YoloV3 running on the server. Another *However, due to encoding, this accuracy drops slightly by around* 0.1% *to* 0.5% *because of the lossy compression of the frames*, as depicted in Fig. 6.4.

Insight: Edge-MAR with H.264 provides a considerable reduction in data size in this experiment, but at the cost of reduced inference accuracy by 0.5%, implying that using H.264, an Edge-MAR system can save around 92% of the allocated bandwidth with slightly reduced accuracy. Neither the data size nor the inference accuracy depends on the CPU frequency. Though frame resolution does not influence encoding, with additional information in a frame, encoded data size and compression ratio increase.

6.3.4 Impact of RSS on transmission latency and energy

The impact of the signal strength of the wireless medium is pivotal in the transmission of data from mobile devices to the server. Both latency and energy consumption due to transmission increase with the decrease in RSS, which in turn increases the overall latency and energy consumption. Three RSS levels are generated for the experiment: -30 dBm, -42 dBm, and -60 dBm. Fig. **??** and Fig. 6.5 show the transmission latency and energy, respectively, for different RSS at different CPU freq. for both Edge-MAR and Edge-MAR with H.264. For RSS$=-60$ dBm, in Edge-MAR and Edge-MAR with H.64, transmission latency increases on average by 44.41 ms and 7.73 ms at 1 GHz, 32.13 ms and 6.38 ms at 2 GHz, and 25.36 ms and 5.08 ms at 3 GHz of CPU freq. sequentially from RSS=−30 dBm. At the same RSS, a corresponding increase in energy consumptions for transmission for these systems are 49.38 mJ and 9.38 mJ at 1 GHz, 89.75 mJ and 4.31 mJ at 2 GHz, and 75.63 mJ and 24.75 mJ at 3 GHz of CPU freq. from −30 dBm. Fig. 6.6 illustrates the difference in transmission latency for encoding from Edge-MAR only for different RSS.

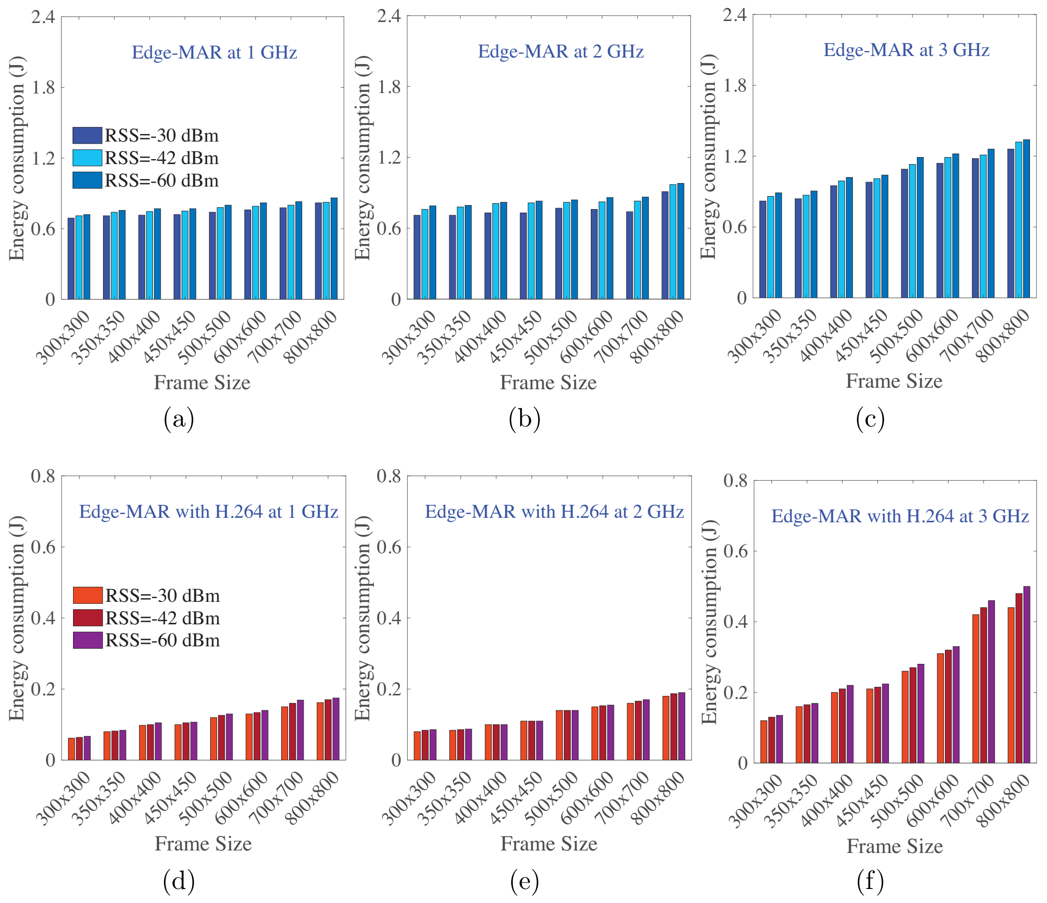
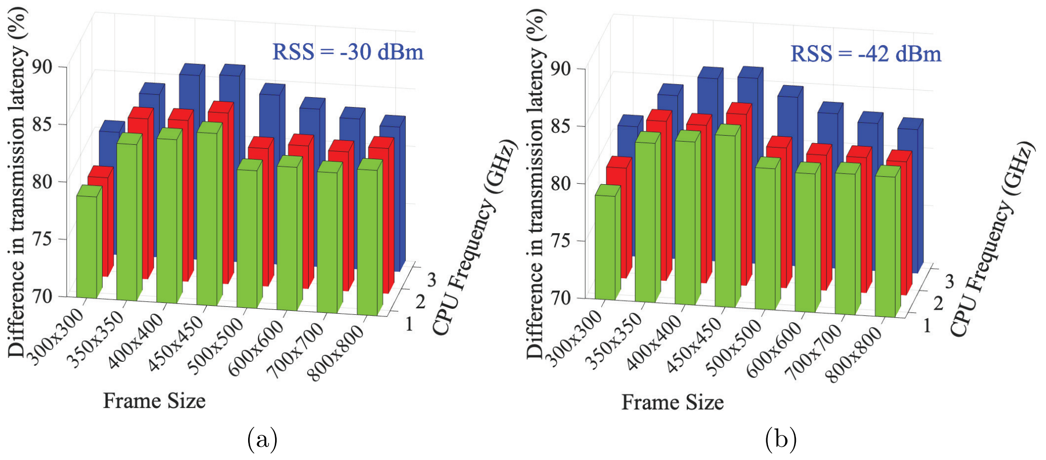


Figure 6.5: Energy consumption for frame transmission for different RSS for Edge-MAR at CPU frequency (a) 1 GHz, (b) 2 GHz, and (c) 3 GHz, and for Edge-MAR with H.264 at (d) 1 GHz, (e) 2 GHz, and (f) 3 GHz.

Insight: It is evident that with the rise in CPU frequencies, transmission latency decreases. However, at 2 GHz CPU frequency, the transmission energy does not increase sharply, compared to the other frequencies, due to smartphone architectures' high compatibility with the 2 GHz range. Moreover, due to encoding, transmission latency reduces by almost 80% from that of only Edge-MAR.



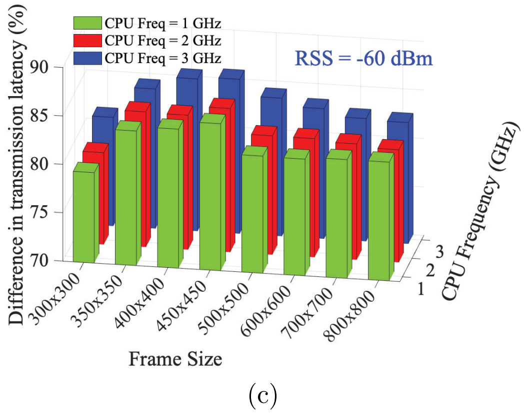


Figure 6.6: Difference in transmission latency due to encoding in percentage for RSS levels (a) −30 dBm, (b) −42 dBm, and (c) −60 dBm.

6.3.5 Impact of encoding on latency and energy consumption

Since the proposed Edge-MAR system involves H.264 at the client side, encoding has an impact on both latency and energy consumption of mobile devices, shown in Fig. 6.7. With the increase in CPU frequency, the encoding latency decreases, but the encoding energy increases. However, at 2 GHz CPU frequency, the reduction in latency is high, and the increase in energy consumption is much lower than that at 3 GHz.

Insight: At 2 GHz of CPU frequency, smartphones provide higher efficiency in terms of both encoding latency and energy consumption due to encoding. For frame size 450×450, it gives the optimal latency and energy consumption.

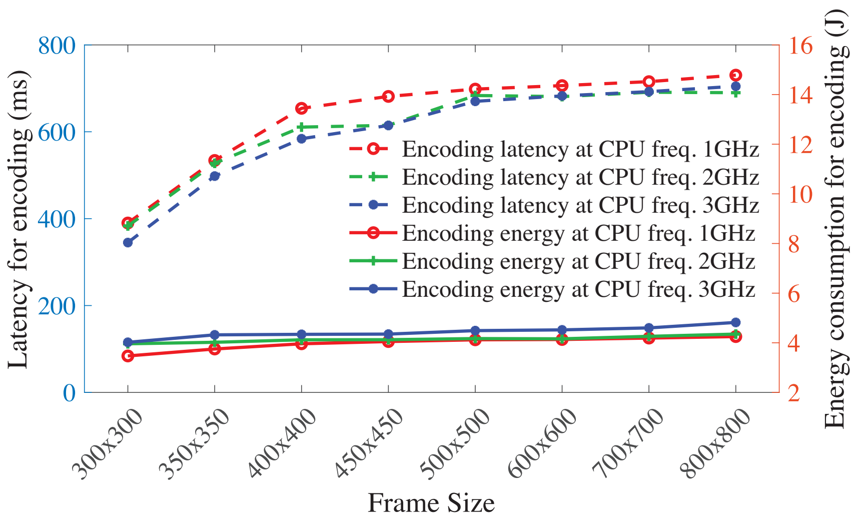


Figure 6.7: Latency and energy consumption due to H.264 encoding.

6.3.6 Regression model

To develop regression-based models, frame size is denoted as S_{frame}, data size as S_{data}, accuracy for encoded frames as Acc_{encode}, difference in overall and transmission latency from Edge-MAR to Edge-MAR with H.264 as $\triangle \mathrm{t}_{all}$ and $\triangle t_{transm}$ respectively, encoding latency as t_{encode}, the difference in overall energy consumption from Edge-MAR to Edge-MAR with H.264 as $\triangle E_{all}$, RSS levels as S_{RSS}, and finally CPU frequency as f. The proposed models for Acc_{encode}, t_{encode}, $\triangle t_{transm}$, and S_{frame} are summarized in Table 6.2. The R^2 values show the strength of the relationship between the model and the dependent variables, implying a good fit of the model. This model can be used to further design network- and energy-aware H.264-based Edge-MAR systems where developers can choose the independent variables to achieve desired values of dependent variables within the 95% confidence boundary.

Table 6.2: Proposed linear regression-based models for MAR systems

Parameters	Proposed models	R^2 value
Acc_{encode}	$0.81 + 2.94\times10^{-5} S_{frame} + 0.34f$	0.73
t_{encode}	$382.77 + 0.53 S_{frame} - 19.89f$	0.64
$\triangle t_{transm}$	$23.08 - 0.21 S_{frame} + 5316.3 S_{data} - 11.8f - 0.89 S_{RSS}$	0.71
S_{frame}	$-5306.5 - 2.33f - 1.18 S_{RSS} - 39.38 \triangle E_{all} + 1.2 \triangle t_{all} + 0.96 t_{encode} - 2.43 t_{transm} + 6075.7 Acc_{encode} + 7508.2 S_{data}$	0.97

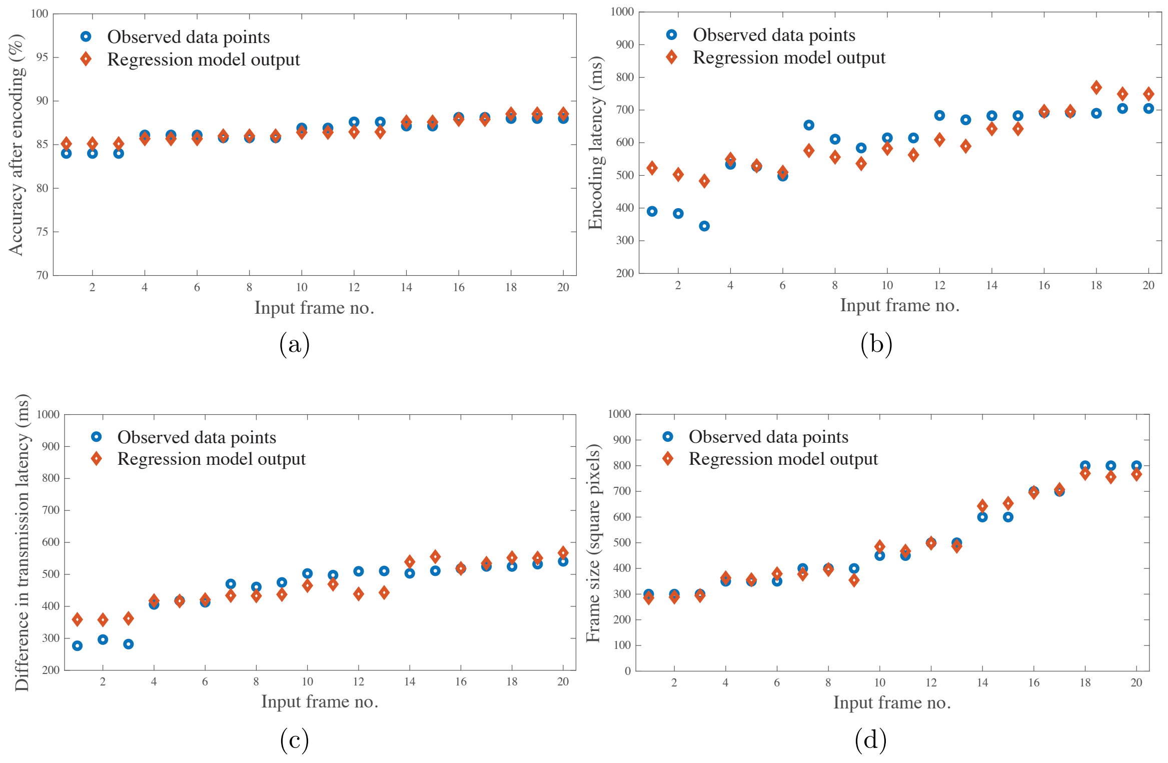


Figure 6.8: Performance of regression models in predicting MAR parameters at consecutive input frames: (a) accuracy after encoding, (b) encoding latency, (c) difference in transmission latency, and (d) frame size.

CHAPTER 7: PROPOSED MOBILE AI QUALITY-OF-SERVICE ENHANCEMENT FRAMEWORK

In this chapter, a novel edge-MAR system based on deep reinforcement learning – 'REAL" (Reinforced Edge Assisted Learning) is proposed, where the edge server is provided with the system states (i.e., the wireless link quality and battery energy level) to make dynamic and smart decisions for mobile devices on whether to offload and how to process the input image frames for the MAR service based on a reward cost function of end-to-end latency, energy consumption, service accuracy, and data size. The learning algorithm for the proposed system is trained online using the system states and provides rewards based on the actions it takes, with the principle objective of collectively optimizing the system's latency and energy consumption of the mobile devices. The design of the proposed framework REAL is described in Section 7.1. The learning algorithm and the training are presented in Section 7.2.

In addition, extensive experiments using the implemented testbed are carried out with different mobile devices and powerful machines under strictly controlled environments so that the experiment is repeatable and replicable by future researchers. The experimental procedure is discussed in detail in Section 7.3. The latency is measured using carefully placed timestamps at both the client and the server side, as well as the energy consumption of the mobile devices using an external power measurement tool. The offloaded data size and service accuracy are measured from within the application. These performance metrics are used for the system performance evaluation (Sec. 7.4). The overall QoS of REAL is observed to be higher than all the other benchmark MAR systems. Another comparison with state-of-the-art offloading techniques proves the superior performance of REAL.

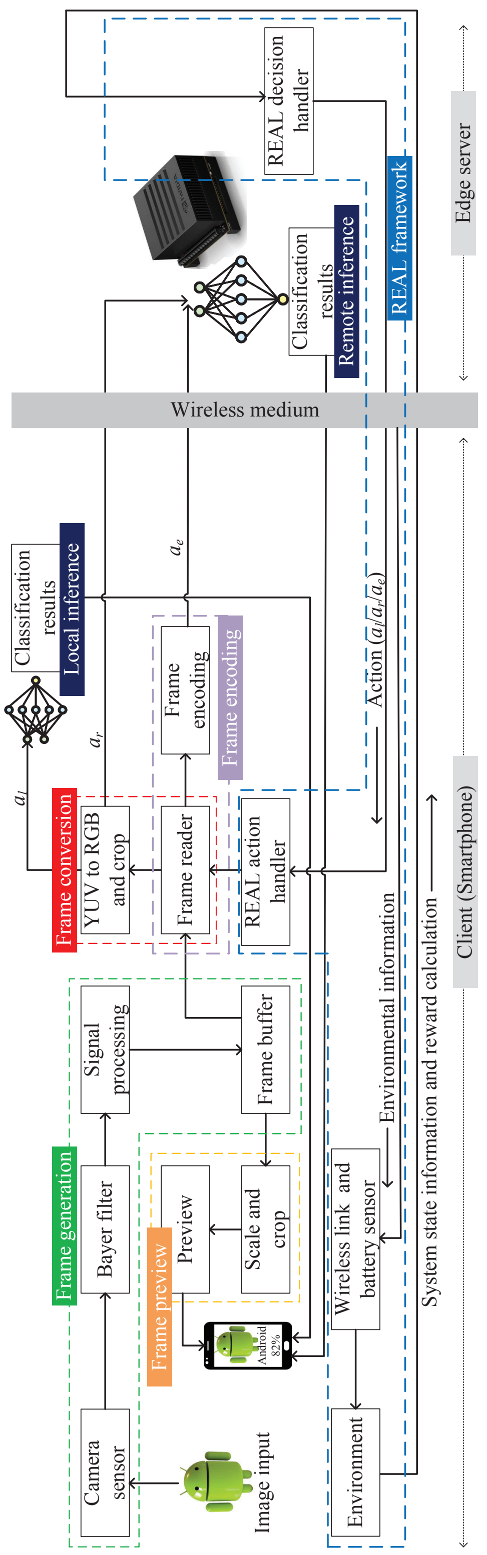


Figure 7.1: Processing pipeline of an object detection application with the proposed REAL smart edge-MAR system.

7.1 Proposed System Architecture: REAL

In this section, the design of the proposed REAL edge-MAR system is presented considering object detection, which is a basic MAR application. The system consists of three major components: a client, an edge server, and a hybrid edge-client REAL framework. The processing pipeline of the REAL edge-MAR system is discussed below as shown in Fig. 7.1.

Client: The client (i.e., a mobile device) processes five major tasks for this application which are listed below.

- First, it starts with *image generation*. Using the camera sensor, the client first captures a frame with objects. Then, the frame goes through a Bayer filter and a digital signal processor. After that, the frame is stored in a frame buffer.
- Second, it goes for *frame preview*. During the *frame preview*, the frame is first scaled and cropped according to the display requirements. Next, the frame is displayed on the screen. Then, depending on the action space from REAL, it goes to any of the two stages: *frame conversion*, *frame encoding*.
- Third, for *frame conversion*, the frame goes to the frame reader first. Then, it is converted to RGB from YUV color format and cropped according to the requirement of the CNN. On the other hand, *frame encoding* encodes the frame using H.264 encoder with specific encoding parameters, such as I-frame intervals and video bitrates.
- Fourth, depending on the decided action, the converted frame may be passed to a local light-weight CNN model, and a detection result is generated, which is the final task called *local inference*. Another option is to send the color-converted frame to the edge server for *remote inference*. Otherwise, if the action is to transmit the encoded frames, *frame conversion* is skipped. These frames are

sent to the edge server via a wireless medium (i.e., Wi-Fi or cellular networks).

- Finally, the results from either *local inference* or *remote inference* are then sent back to the *frame preview* to be displayed together with the previewed frame.

Edge server: The edge server (i.e., a powerful computing machine) executes *remote inference* with a larger and deeper CNN model. Depending on the actions decided by REAL, the color-converted frame from *frame conversion* or the encoded frames from *frame encoding* tasks are received from the client for *remote inference*. The detection results are sent to the client's *frame preview* to be displayed together with the previewed frame on the screen of the client.

REAL framework: REAL is a hybrid client-edge-based smart decision system. The workflow of the framework is listed as follows.

- In the proposed framework, whenever the camera sensor captures a frame, the client first collects the wireless link quality data using the *wireless link sensor* from the wireless medium. Then, using the *battery sensor*, it collects the current energy level data. Afterward, the client sends this system state information to the edge server.

- The edge server generates action decisions using the *REAL decision handler* based on the optimal policy.

- The decision handler sends the decisions back to the client's *REAL action handler*. This action handler provides input to the frame reader, which then executes necessary actions. These processes are completed by the time a frame reaches the frame reader at the client side.

- After the completion of object detection on a single frame, the client's *environment* calculates the reward and sends it back to the edge server.

- The server recalculates the reward using a discount rate and updates its policy.

A detailed description of REAL with the algorithm and utility function is provided in the next section.

7.2 REAL: Learning Algorithms and Training

In this section, the use of deep reinforcement learning is justified to solve the research problems discussed earlier (Section 7.2.1), the utility function is defined (Section 7.2.3), and the learning algorithm (Section 7.2.5) and the training process (Section 7.2.6) are described.

7.2.1 Choosing the right learning

Introducing REAL in an edge-MAR system makes the whole process smart and dynamic to deal with the varying wireless environments and limited energy constraints of mobile devices. It takes the wireless link quality and the energy level of a mobile device as the system state input and takes actions based on the states. The system state, action space, reward function, and discount rate are denoted here as follows:

- System states, $\mathbb{S} = s_{(i,j)_t}$, is a continuous space, where $i \in \mathbb{R}$, is the wireless RSSI value, and $j \in [0, 100]$, is the energy level (i.e., the remaining battery capacity) of the device at time t.

- Action space, $\mathcal{A} = a_{n_t}$, is a discrete set of actions at time, t, where $n = 1, 2, ..., N$. Here, N denotes the total number of actions, including local inference, remote inference, and remote inference with encoding (with a combination of encoding parameters that need to be controlled, e.g., I-frame and P-frame intervals, video bitrate, frame rate, and resolution). For simplicity, the actions are classified in broader categories of local inference action (a_l), remote inference action with color-converted frames (a_r), and remote inference action with encoded frames (a_e), where a_e contains all the elements of set $\mathcal{A}$ with encoding parameters such as I- and P-frame intervals, video bitrate, frame rate, resolution, and profile.

- State transition probability matrix, $\mathbb{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s'_{(i,j)} | S_t = s_{(i,j)}, A_t = a_{n_t}]$. Here, the state transitions are quite uncertain, with a lot of possible transition targets. For example, if a device is highly mobile during a time period, the RSSI level can vary over a broad range. Additionally, the energy level may experience non-linear degradation due to computation resource allocation, hardware configurations, RSSI, and battery health (state-of-health). Moreover, the energy level may also rise while getting recharged.

- Reward function, $\mathbb{R}^a_s = \mathbb{E}[R_{t+1} = r_t | S_t = s_{(i,j)_t}, A_t = a_{n_t}]$, is calculated from the latency and energy consumption at time, t, where $r_t \in [-1, 1]$.

- Discount rate is $\gamma \in [0, 1]$. The calculated reward is discounted toward the policy update according to this rate, γ.

In short, the edge-MAR system appears to have infinite combinations of states where the actions do not guide the next state. Therefore, it is a *stochastic environment with a discrete set of actions.* Since the state transition is too complex, the use of any heuristic optimization algorithm cannot solve the research problem of optimizing the overall latency, energy, accuracy, and data size. Furthermore, the system runs in a real-time manner, which requires *online recurrent training.* This training should be *off-policy* to reduce the training latency since off-policy algorithms are good for parallel processing and continuous exploration of suitable actions [145]. A specific deep reinforcement learning tool – soft actor-critic (SAC) algorithm [146] can be helpful in this research that assumes a stochastic environment with a continuous action space. However, the action space considered in this research is discrete, i.e., it has a limited number of actions. As a result, REAL needs a modified SAC algorithm, namely *discrete-SAC* [147]. This DRL network takes latency and energy data to calculate the reward for each action. Hence, it is necessary to define the latency and energy calculation.

7.2.2 Latency and energy calculation

The processing pipeline of REAL has several different components. The latency and the energy models comprise of all the delays and energy consumption for the individual tasks explained in Section 7.1. In addition, the latency and energy calculations are different based on the actions taken according to REAL. Therefore, the total latency for the n^{th} frame is

$$L_{tot}^n = \begin{cases} L_{gen}^n + L_{con}^n + L_{pre}^n + L_{loc}^n, & a_t = a_1, \\ L_{gen}^n + L_{con}^n + L_{pre}^n + L_{off}^n + L_{rem}^n, & a_t = a_2, \\ L_{gen}^n + L_{enc}^n + L_{pre}^n + L_{off}^n + L_{rem}^n, & a_t = a_3, \end{cases} \tag{7.1}$$

where L_{gen}, L_{con}, L_{enc}, L_{pre}, L_{loc}, L_{off}, and L_{rem} represent the latency for frame generation, conversion, encoding, preview, local inference, frame offloading, and remote inference, respectively. With this model, the client's total energy consumption model for the n^{th} frame can be developed as

$$\begin{aligned} E_{tot}^n &= \Big(\int_0^{L_{gen}^n} P_{gen}^n \, dt + \int_0^{L_{con}^n} P_{con}^n \, dt + \int_0^{L_{enc}^n} P_{enc}^n \, dt \\ &\quad + \int_0^{L_{off}^n} P_{off}^n \, dt + \int_0^{L_{loc}^n} P_{loc}^n \, dt + \int_0^{L_{rem}^n} P_{rem}^n \, dt \\ &\quad + \int_0^{L_{pre}^n} P_{pre}^n \, dt + \int_0^{L_{tot}^n} P_{base}^n \, dt \Big) \cdot L_{tot}^n, \end{aligned} \tag{7.2}$$

where E_{tot}, P_{gen}, P_{con}, P_{enc}, P_{pre}, P_{off}, P_{loc}, and P_{rem} are the client's total energy consumption, power consumed during frame generation, conversion, encoding, preview, offloading, local inference, and remote inference. There is another power consumption that contributes to the total energy – base power (P_{base}), which is always consumed during an application due to the minimal background activities in a mobile device.

7.2.3 Definition of QoS (The utility function)

In this chapter, the QoS of an MAR application is defined as a function of end-to-end latency (L_{tot}), energy consumption (E_{tot}), mean service accuracy (A), and offloaded data size (S_{off}) for frame n. This leads to the reward cost function, R_c as given below

$$R_c = \begin{cases} C_{l_1} L_{tot} + C_{e_1} E_{tot} - C_a A + C_{off} S_{off}, & E_B \geq E_{B_{th}}, \\ C_{l_2} L_{tot} + C_{e_2} E_{tot} - C_a A + C_{off} S_{off}, & E_B < E_{B_{th}}, \end{cases} \tag{7.3}$$

where C_a, C_{off}, E_B, and $E_{B_{th}}$ are coefficients of accuracy and offloaded data size, battery energy level, and threshold of battery energy level. When E_B falls below $E_{B_{th}}$, coefficients of latency and energy C_{l_1} and C_{e_1} are changed to C_{l_2} ($< C_{l_1}$) and C_{e_2} ($> C_{e_1}$). As more weight is put on energy consumption in the reward cost function, REAL focuses more on saving energy during low battery energy levels, which is essential to prolong the battery life of a mobile device during an MAR application. *The lower the value of R_c is, the higher QoS of the MAR system is.*

7.2.4 Reward calculation

The reward for actions is calculated based on the total latency and energy consumption of the client. Reward, r_t, at time t can be expressed as

$$r_t = \begin{cases} +1, & \text{when } R_c \leq R_{c_{th}}, \\ 0, & \text{when } R_c > R_{c_{th}} \quad \& \quad L_{tot} \leq L_{th} \& E_{tot} \leq E_{th}, \\ -1, & else, \end{cases} \tag{7.4}$$

where $R_{c_{th}}$, L_{th}, and E_{th} are threshold values of the reward cost, total latency, and energy consumption, respectively. These values are pre-set using experimental data for a few discrete system states. However, $R_{c_{th}}$, L_{th} and E_{th} are subject to frequent updates. When a lesser total is detected than the threshold, the value is stored as the threshold for a specific state, $s_{i,j}$, based on Algorithm 2.

Algorithm 2 The pseudo-code of REAL reward calculation.

```
Input: $R_c$, $L_{tot}$, and $E_{tot}$ at $t$, and $R_{c_{th}}$, $L_{th}$, and $E_{th}$ for $s_{i,j}$.
Begin
  Initialize Reward buffer, $R = \emptyset$, reward, $r_t = \emptyset$, threshold buffers, $R_{c_{th}}$, $L_{TH}$ and $E_{TH}$ with pre-set experimental data
  foreach $t$ do
    if $R_c \leq R_{c_{th}(i,j)}$ then
      $r_t = +1$
      $R_{c_{th}(i,j)} \leftarrow R_c$                          ▷ Update $R_{c_{th}}$
      $L_{th(i,j)} \leftarrow L_{tot}$                          ▷ Update $L_{th}$
      $E_{th(i,j)} \leftarrow E_{tot}$                          ▷ Update $E_{th}$
    end
    else if $R_c > R_{c_{th}(i,j)}$ & $L_{tot} \leq L_{th(i,j)} \& E_{tot} \leq E_{th(i,j)}$ then
      $r_t = 0$
    end
    else
      $r_t = -1$
    end
    $R[t] \leftarrow r_t$                                       ▷ Store in $R$
    return $r_t$, $R_{c_{th}(i,j)}$, $L_{th(i,j)}$, and $E_{th(i,j)}$
end
Output: Reward $r_t$ at $t$, updated $R_{c_{th}}$, $L_{th}$, $E_{th}$ for $s_{i,j}$.
```

7.2.5 Overview of the learning algorithm of REAL

REAL is a discrete-SAC-based process. The critic network has a value function given as

$$V(s_{(i,j)_t}) = \pi(s^T_{(i,j)_t}) \left[Q(s_{(i,j)_t}) - \alpha \log(\pi(s_{(i,j)_t}))\right], \tag{7.5}$$

where T, $\pi(s)$, $Q(s)$, and α are total timesteps, policy function, Q-function, and temperature parameter, respectively. The objective function of the temperature parameter is

$$J(\alpha) = \pi(s^T_{(i,j)_t}) \left[-\alpha \log(\pi(s_{(i,j)_t})) + \overline{H}\right], \tag{7.6}$$

where $\overline{H}$ represents the target entropy, which depicts the level of exploration of actions. Next, the new objective function for the policy, π becomes

$$J_\pi(\phi) = \mathbb{E}_{s_{(i,j)_t}\sim D}\left[\pi(s^T_{(i,j)_t})[\alpha \log(\pi_\phi(s_{(i,j)_t})) - Q_\theta(s_{(i,j)_t})]\right], \tag{7.7}$$

where D is the replay buffer of past experiences consisting of current state $s_{(i,j)_t}$, current action a_t, current reward r_t, and the next state $s_{(i,j)_{t+1}}$. Finally, the soft-Q function to be trained is

$$\begin{aligned} J_Q(\theta) = \mathbb{E}_{(s_{(i,j)_t},a_t)\sim D}&\left[\frac{1}{2}(Q_\theta)(s_{(i,j)_t}, a_t)\right. \\ &\left.-(r + \gamma \mathbb{E}_{s_{(i,j)_{t+1}}\sim p(s_{(i,j)_t},a_t)}[V_{\bar{\theta}}(s_{(i,j)_{t+1}})])^2\right], \end{aligned} \tag{7.8}$$

where γ and $V_{\bar{\theta}}(s_{(i,j)_{t+1}})$ are the discount rate and the target network value function, respectively. To avoid over-estimation, two Q-parameters are introduced in the network: θ_1 and θ_2.

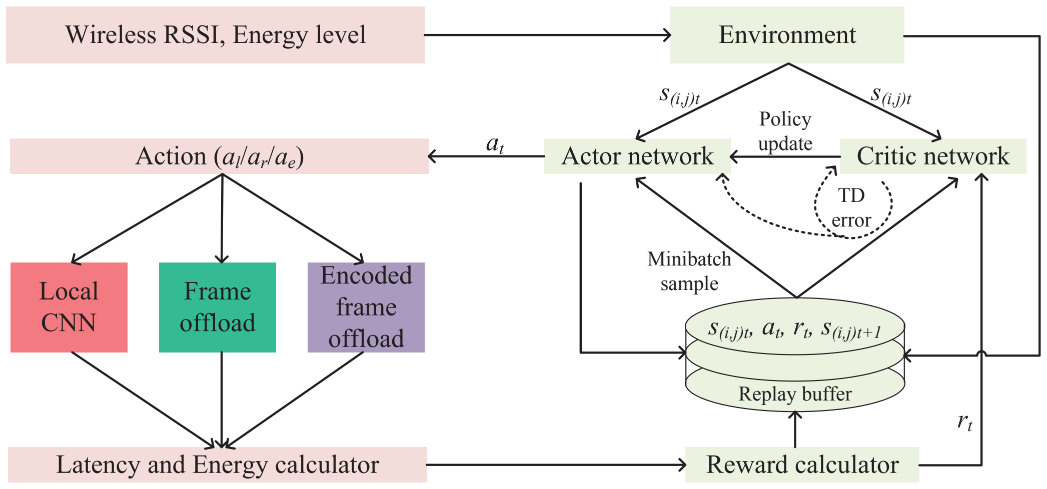


Figure 7.2: Overview of the proposed REAL smart edge-MAR system.

The overview of REAL is shown in Fig. 7.2. At time t, the environment first collects the system states $s_{(i,j)_t}$, and sends them to both the actor and critic network. Based on the policy object function from (7.7), the actor network generates action, a_t

from the action space, $\mathcal{A}$. This action is then sent to the REAL action handler at the client. After completing the action, the client calculates the total latency and energy consumption, then sends these data to the reward calculator along with the mean accuracy and offloaded data size. Calculated by (7.4), reward, r_t is first multiplied by the discount rate, and then sent to both the critic network and the replay buffer, D. From the minibatch samples from D, both the actor and critic network get past experiences, $s_{(i,j)_t}, a_t, r_t, s_{(i,j)_{t+1}}$. Temporal difference (TD) error is calculated by the critic, and an updated policy is shared with the actor network. Future actions are adjusted based on this updated policy.

7.2.6 Training of REAL

REAL is trained for 100 episodes with 200 frames per episode. The learning rate is carefully chosen to avoid the classic over-fitting and under-fitting problems in the machine learning paradigm. The proposed learning model is tested with five different learning rates, α: 0.01, 0.001, 0.0005, 0.0001, and 0.00001. Fig. 7.3 shows the convergence pattern for different learning rates. Since each episode contains 200 frames, before applying the discount rate, γ, the maximum possible reward at each episode is 200, according to the reward definition in this chapter. It is evident that the training with α values of 0.0001 and 0.00001 reaches desired rewards (around 80% of the maximum possible reward) much sooner than with other learning rates. However, to maintain satisfactory QoS of a real-time MAR application, the overall latency of the learning algorithm should be considered, as well.

The total latency of a reinforcement learning model consists of exploration, exploitation, and execution delays [148]. Fig. 7.4 shows such latency comparison for different learning rates of REAL. Choosing a value of α lower than 0.0001 causes a higher delay, which can cause lower overall QoS for an MAR application. In addition to this, the total latency of $\alpha = 0.0001$ is similar to the higher values, which motivates us to select this as the learning rate to train the proposed system. The significant

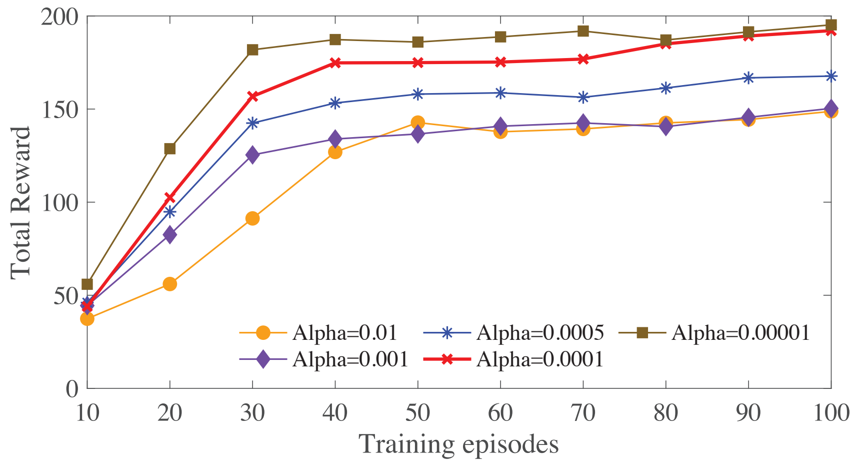


Figure 7.3: Convergence at different learning rates (α) of the learning algorithm of REAL (before applying discount rate, γ).

hyperparameter values and types are listed in Table 7.1.

The overall training performance of REAL is shown in Fig. 7.5. As the number of episodes reaches around 85, the total rewards per episode get stable at around 190 with a mean reward of 78%, which is satisfactory for the proposed system. The term “epoch” is not used here as each individual training episode may have different data depending on several uncontrollable environmental parameters such as hardware acceleration, CPU processing, and slight changes in lights.

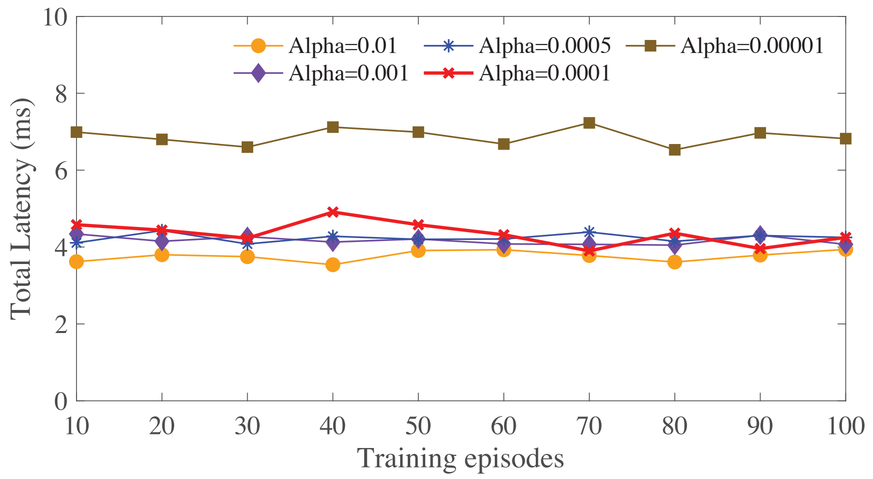


Figure 7.4: Total latency at different learning rates of the learning algorithm of REAL.

Table 7.1: Hyperparameters used for the training of the learning algorithm of REAL

Hyperparameter	**Value/Type**
Layers	2 convolutional layers and 1 fully connected layer
Convolutional channels per layer	[32, 64, 64]
Fully connected layer hidden units	3
Batch size	64
Replay buffer size	5000
Discount rate	0.99
Learning rate	0.0001
Optimizer	Adam
Loss	Mean squared error

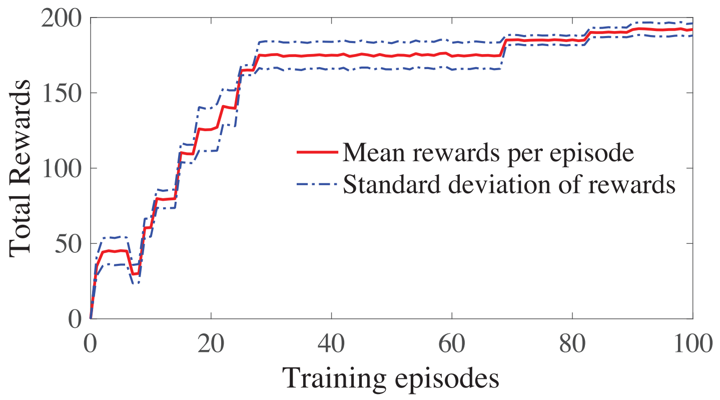


Figure 7.5: Training performance (in rewards) of the learning algorithm of REAL at learning rate, $\alpha = 0.0001$ (before applying discount rate, γ).

7.3 Experimental Testbed and Methodology

Testbed implementation: Copious experiments are carried out using the implemented testbed. The devices used in the experiment are listed in Table 7.2. Different mobile devices of the latest releases from unique manufacturers with diverse configurations (clients 1-6) are used as the client devices, including one high-performing computing machine (client 7) for preliminary study. The most powerful device listed here is the Jetson AGX Xavier [140], having an 8-core ARM 64-bit CPU with 32 GB 256-bit LPDDR4x 137 GB/s memory and 512-core Volta GPU with tensor cores, which is used as the edge server. A LinkSys dual-band router (2.4 GHz and 5 GHz) [141] is used as the wireless medium, which connects the clients and the server via Wi-Fi. An external power measurement tool – Monsoon Power Monitor [142] is connected to mobile devices for energy consumption measurement purposes. However, only clients 1-3 are considered for this energy measurement, as mobile devices with batteries (clients 4-6) are needed to test the proposed system completely because devices without batteries cannot provide the changing battery level data, which is essential to creating the dynamic system states.

Methodology: The local inference of object detection is executed with the `MobileNetv1.0` model [149] optimized by `TensorFlowLite` [150] having 64 hidden layers with an input size of 300. For the remote inference with converted frames, `YOLOv3` [143, 151] with 106 layers is used for remote inference on the edge server, which uses `COCO` dataset with 80 different classes of objects [144]. The 2.4 GHz band of the router is used to access the Wi-Fi network. To create different RSSI levels, the mobile devices are moved away from the Wi-Fi access point at a constant speed to a constant direction with line-of-sight considerations. For this experiment, three actions from the action space, $\mathcal{A}$ are implemented only, which are: local inference (a_l), remote inference with color-converted frames (a_r), and remote inference with encoded frames (a_e) having the encoding configuration parameters I-frame interval 5, and maximum video bit-rate 30 MB/s. To encode the frames, an open-source compression algorithm – OpenH264 [152] is applied, which uses H.264 encoding to compress video frames that provides around 140% compression rate in the experiments. For all the experiments, the frame rate is set to 30 fps, CPU frequency to 2 GHz, and captured input frame size at 500×500 pixels, which is studied as optimal parameters for edge-MAR systems [19]. Android Studio is used to develop the entire client-side application with Java for Android OS-based phones and AR glass. Remote inference and REAL applications are written in Python.

Energy consumption measurement requires connectivity between the power mon-

Table 7.2: Brief specifications of the devices used in the experiments for mobile AI QoS enhancement

Denotation	Model	System-on-Chip	CPU	GPU	RAM	OS	Wi-Fi	Release Date
Client 1	Huawei Mate 40 Pro	Kirin 9000 (5 nm)	Octa-core (1×3.13GHzA77 3×2.54GHzA77 4×2.05GHzA55)	Mali G78	8GB LPDDR5	Android 10	802.11 a/b/g/ n/ac/ax	October, 2020
Client 2	OnePlus 8 Pro	Snapdragon 865 (7 nm)	Octa-core (1×2.84GHz 3×2.42GHz 4×1.8GHz Kryo 585)	Adreno 650	8GB LPDDR5	Android 10	802.11 a/b/g/ n/ac/ax	April, 2020
Client 3	Motorola One Macro	Helio P70 (12 nm)	Octa-core (4×2.0GHzA73 4×2.0GHzA53)	Mali G72	4GB LPDDR4X	Android 9	802.11 b/g/n	October, 2019
Client 4	Vivo IQOO Z1	Mediatek MT6889Z (7 nm)	Octa-core (4×2.6GHzA77 4×2GHzA55)	Mali G77	6GB LPDDR4X	Android 10	802.11 a/b/g/ n/ac	May, 2020
Client 5	Google Pixel 4a	Snapdragon 730G (8 nm)	Octa-core (2 × 2.2GHz Kryo470G 6 × 1.8GHz Kryo470S)	Adreno 618	6GB LPDDR4X	Android 10	802.11 a/b/g/ n/ac	August, 2020
Client 6	Google Glass Enterprise Edition 2	Snapdragon XR1	Octa-Core Kryo (2×2.52GHz 6×1.7GHz)	Adreno 615	3GB LPDDR4	Android 8.1	802.11 a/b/g/ n/ac	May, 2019
Client 7	Nvidia Jetson TX2	Tegra	Dual-Core NVIDIA Denver2 Quad-Core A57 MPCore	256-core NVIDIA Pascal	8GB LPDDR4	Ubuntu 18.04	–	March, 2017
Edge server	Nvidia Jetson AGX Xavier	Tegra	Octa-core ARM v8.2	512-core Volta Tensor Cores	32GB LPDDR4X	Ubuntu 18.04 LTS aarch64	–	October, 2018

itor and mobile devices. Batteries are removed by applying heat to the back of the phones. The power terminals are then soldered to wire extensions. These wires are connected to the power monitors external probes. The power monitor provides energy to the mobile devices and simultaneously measures the power consumption every 2 ms. As mentioned before, there is a base power in every mobile device, which needs to be minimized as much as possible to get the most precise measurement. Hence, all the irrelevant background applications and features are closed, camera features are disabled, the display brightness is kept at the lowest level, and wireless connectivities other than Wi-Fi are shut down. To conduct the experiments in a closed and controlled environment, all the measurement data are collected during the night with constant light exposures. To test the object detection application, `COCO test dataset 2017` are used. All the latency and energy consumption data are stored in log files, which later are analyzed to evaluate the performance of REAL and compared with those from other baseline MAR systems.

7.4 Performance Evaluation of REAL

7.4.1 Key Performance Metric

For the proposed REAL edge-MAR system, the average end-to-end latency and average energy consumed per frame by the client devices are considered to be the vital performance metrics since the objective of REAL is to reduce latency and energy consumption under different circumstances. In addition to these, the mean accuracy of object detection and offloaded data size are considered as other key performance metrics. A benchmark accuracy is essential to maintain in most MAR applications. Moreover, the offloaded data size represents wireless network resource utilization by the edge-MAR system. The QoS of an MAR system is calculated as a weighted sum of these four key performance metrics, as explained in Section 7.2.3.

7.4.2 Overall Latency and Energy Consumption

Experiments are performed on baseline MAR systems to compare with the proposed REAL edge-MAR system's performance, where baseline-1, baseline-2, and baseline-

3 denote the local inference only, remote inference only, and remote inference with encoding only, respectively. The system states are varied over a broad range – RSSI levels between the Wi-Fi access point and client devices from 0 dBm to -90 dBm and devices' battery energy level $(1 - DoD)$ from 100% to 10%. The overall latency and energy consumption for the individual components in the processing pipeline of the baseline MAR systems at RSSI < -30 dBm are shown in Fig. 7.6 averaged over 200 frames.

The "local inference" MAR system (baseline-1) executes the CNN model for object detection locally on-device, whereas "remote inference" (baseline-2) and "remote inference with encoding" (baseline-3) execute it on the edge server remotely, which is why these two systems offload frames to the edge. Additionally, "remote inference with encoding" deploys an H.264 encoder at the client side and a decoder at the server side to encode/decode the frames. Therefore, these three systems have different components in their pipelines. In "local inference", the significant delay is caused by the frame conversion and inference. On the other hand, "remote inference" has the maximum delay from offloading frames. However, due to encoding, the frame offloading has a much smaller delay under "remote inference with encoding," but the system has a high delay from encoding itself. The critical takeaway from this study is that in good network conditions, the edge-MAR with encoding has 2% and 20% less total latency as compared to local and remote inference systems, respectively.

The energy consumption of the three systems is also unique in nature. The most energy is consumed in local and remote inference systems by frame generation. However, encoding consumes the maximum energy in remote inference with encoding. Offloading takes much lesser energy in remote inference with encoding than in remote inference. In all the systems, frame preview consumes about 15% of the total energy. Moreover, there is a base energy consumption in all the systems. The overall energy consumption of the encoding-based edge-MAR system surpasses that of the

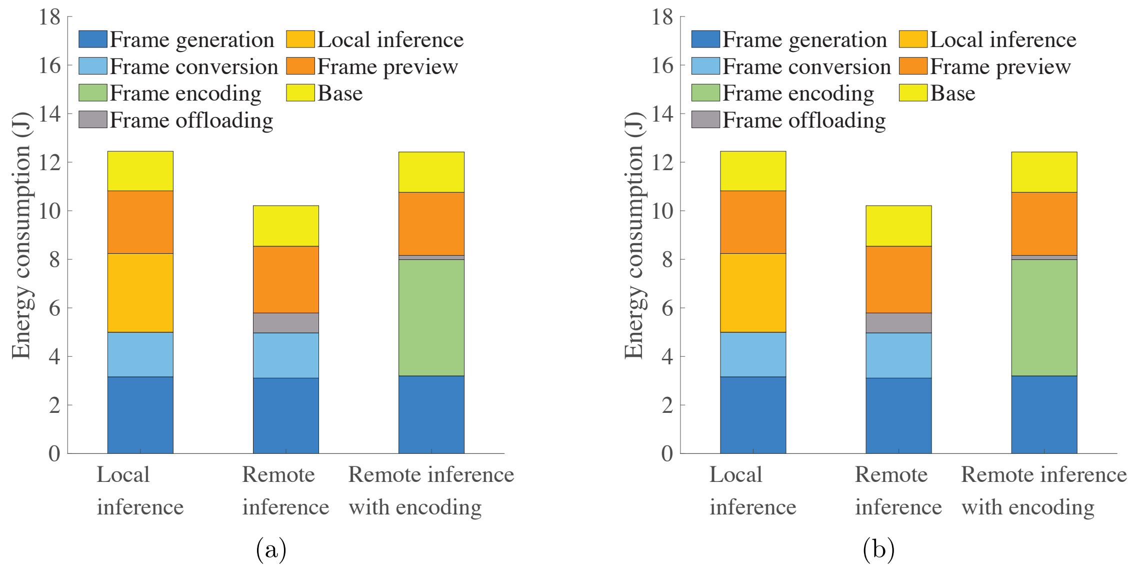
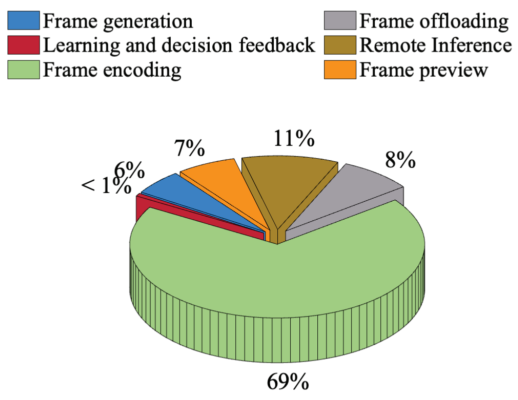


Figure 7.6: The overall average (a) latency and (b) energy consumption of baseline MAR systems at RSSI < -30 dBm.

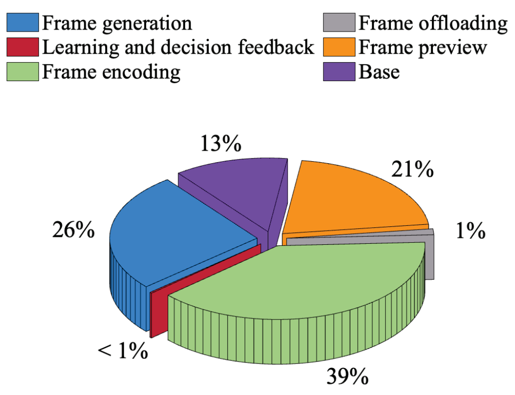
others. *The least amount of energy is consumed by the remote inference system.*

REAL edge-MAR system involves a deep reinforcement learning-based smart edge-MAR decision process. Fig. 7.7 shows a sample REAL edge-MAR system's individual components and their latency and energy consumption at RSSI < -30 dBm, where it adopts action a_e (remote inference with encoding). The training, learning, and decision feedback segment of REAL takes $< 1\%$ delay and energy of the total latency and energy consumption of the mobile device. From this evidence, it can be concluded that REAL can be implemented in an edge-MAR system without adding significant load or causing crucial degradation of QoS.

The total latency and energy consumption of the two remote inference-based systems vary with RSSI levels. Fig. 7.8 depicts that the remote inference system gives rise highest to the latency in all RSSI levels at 80% battery energy level. Offloading is the only part in the processing pipeline of an edge-MAR system that is affected by the degraded wireless link. However, due to the compressed data size, the offloading latency does not vary much in varying RSSI for remote inference with encoding. On the contrary, the energy consumption is higher with encoding due to the compression,

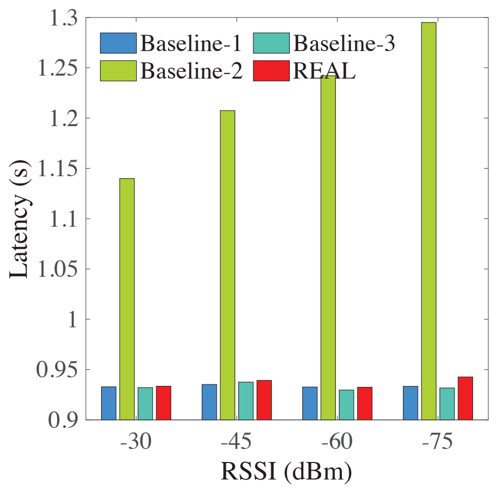


(a)

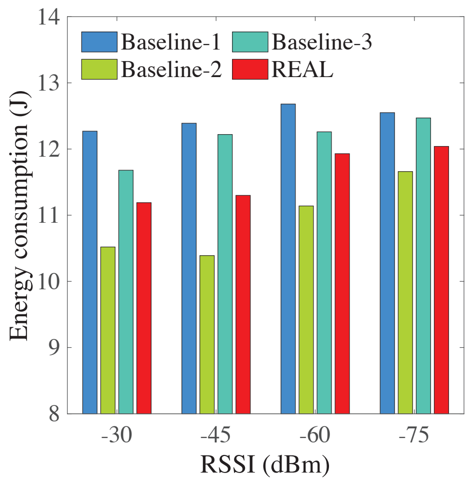


(b)

Figure 7.7: Average percentage breakdown of (a) latency and (b) energy consumption of individual segments of REAL at RSSI < -30 dBm where REAL adopts a_e.



(a)

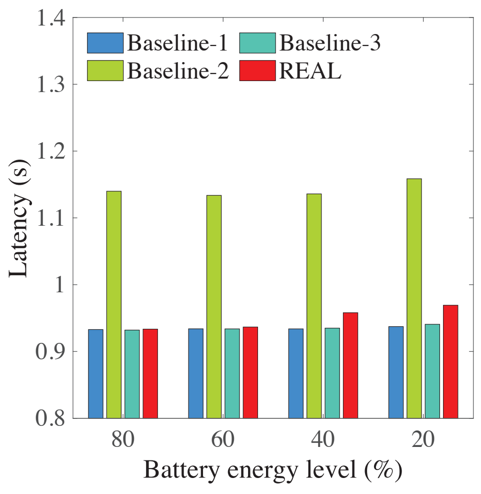


(b)

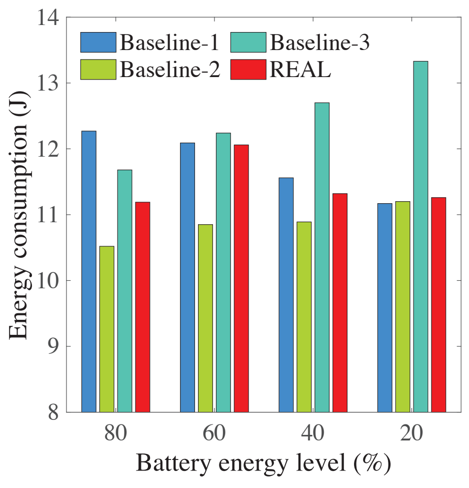
Figure 7.8: Total (a) latency and (b) energy consumption at different RSSI and battery energy level 80%.

and the lowest with the remote inference system, but the latter varies a lot with RSSI. The local inference system does not change with the RSSI levels as it does not involve any transmission over the wireless network. REAL edge-MAR adopts a system among the three abovementioned MAR systems depending on the RSSI and energy levels to increase the QoS. The mean latency of REAL edge-MAR is around 11% lower than the other systems at different RSSI of the wireless link. Additionally, the energy consumption of REAL edge-MAR is 5% lower than that of the baseline systems on average under changing energy levels. Fig. 7.9 shows a similar performance to REAL, where it optimizes both the latency and energy consumption collectively at different battery energy levels and RSSI < -30 dBm. All the performance results of REAL edge-MAR are shown as an average of 100 training episodes with 200 frames.

Highlights: Frame conversion produces a considerable amount of delay in both local inference and remote inference systems. *The highest delay is caused by encoding in remote inference with encoding, but the offloading latency goes down due to the compressed data size.* Frame generation and preview consume a significant amount of energy in all the systems. However, encoding takes a surge of energy but reduces a significant amount while offloading. *REAL edge-MAR adopts a system among these three based on the RSSI and battery energy level so that the mean latency and energy consumption of REAL edge-MAR are optimized dynamically.* The deep reinforcement learning in REAL edge-MAR causes $< 1\%$ latency and energy of the total consumption, which makes it a feasible solution for edge-MAR applications without degrading the QoS.

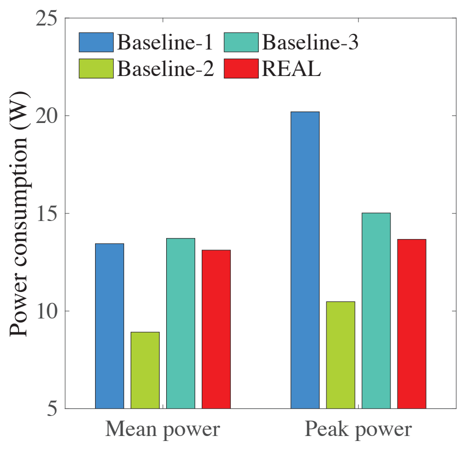


(a)

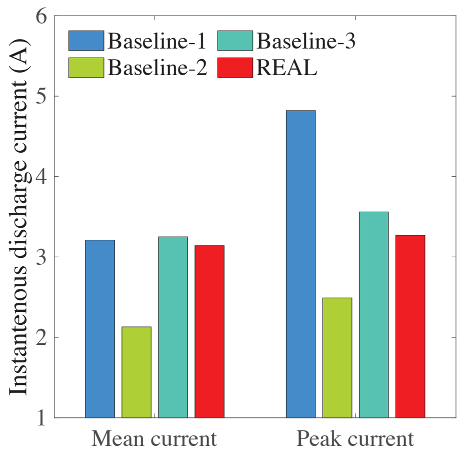


(b)

Figure 7.9: Total (a) latency and (b) energy consumption at different energy levels and RSSI < -30 dBm.



(a)



(b)

Figure 7.10: Mean and peak (a) power consumption and (b) discharge current for different systems at RSSI < -30 dBm and battery energy level 100%.

7.4.3 Power Consumption and Discharge Current Analysis

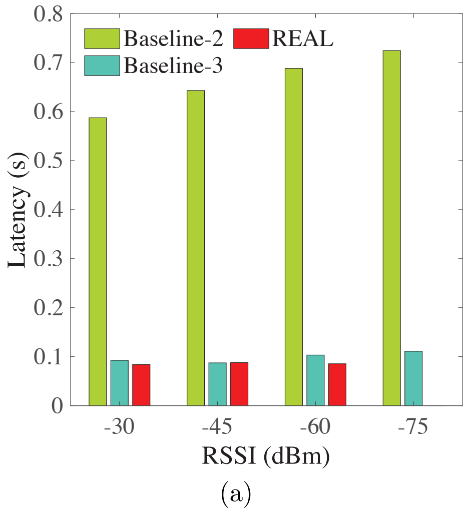
Mobile devices' battery health immensely depends on power consumption and the discharge current. Energy consumption analysis gives more comprehensive insights into the delay and power, whereas instantaneous power and current provide knowledge on how soon the state-of-health degrades due to thermal dissipation [153]. Moreover, discharging current dictates the aging of a li-ion battery [154]. In Fig. 7.10, the mean and peak power consumption and discharge current data are shown for the baseline systems and the proposed framework, REAL.

For baseline-1, the mean power consumption is lower than that of baseline-3. However, Fig. 7.10a shows that the peak power consumption of the baseline-1 system is the highest among all of them. This sudden rise in power consumption can harm the battery's health by dissipating high heat [42], causing user dissatisfaction. A similar trend is observed in the discharge current data (Fig. 7.10b) since the voltage remains stable during the experiments.

Highlights: Power consumption and discharge current are important parameters to consider in mobile augmented reality as these are directly related to the state-of-health of the battery. *Local inference of MAR applications causes a sudden increase in current discharge and power, which is not observed in remote inference systems.* This high peak power and current can lead to faster degradation of mobile devices' energy sources.

7.4.4 Latency and Energy due to Offloading and REAL

Experiments show that offloading is the only component of the entire edge-MAR pipeline affected by varying wireless link quality. Fig. 7.11 exhibits such variations at different RSSI and 80% energy levels. Remote inference with encoding involves much lower offloading latency and energy due to the smaller offloaded data size. This behavior is also shown by Fig. 7.12 at different energy levels and RSSI < -30 dBm. However, since REAL edge-MAR actions suggest empowering different MAR systems at different conditions, both latency and energy consumption are reduced at all levels. The proposed system reduces the offloading latency and energy consumption by around 116% and 83%, respectively, on average, compared to the baseline systems. REAL is observed to have less latency and energy than baseline-2 and baseline-3 systems since it sometimes adopts local inference only (a_l) depending on the net-



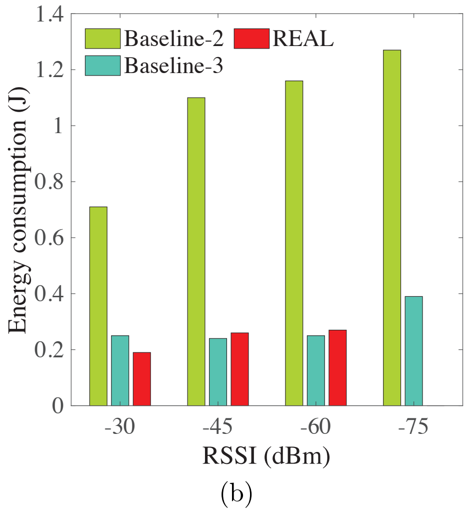


Figure 7.11: Offloading (a) latency and (b) energy consumption at different RSSI and battery energy level 80%.

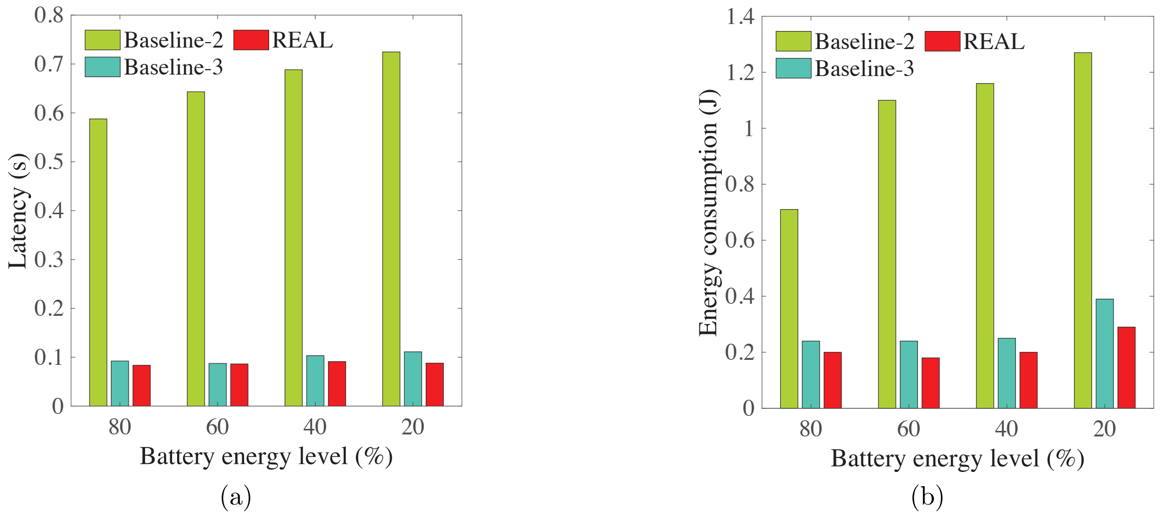


Figure 7.12: Offloading (a) latency and (b) energy consumption different battery energy level and RSSI < -30 dBm.

work conditions, which does not offload at all, resulting in reduced mean latency and energy.

Since the reinforcement learning segment of the REAL edge-MAR system involves transmitting state, action, and reward information over the wireless network, it also encounters such varying natures of latency and energy consumption no matter how smaller the variations are. According to the experiments, as the RSSI degrades from

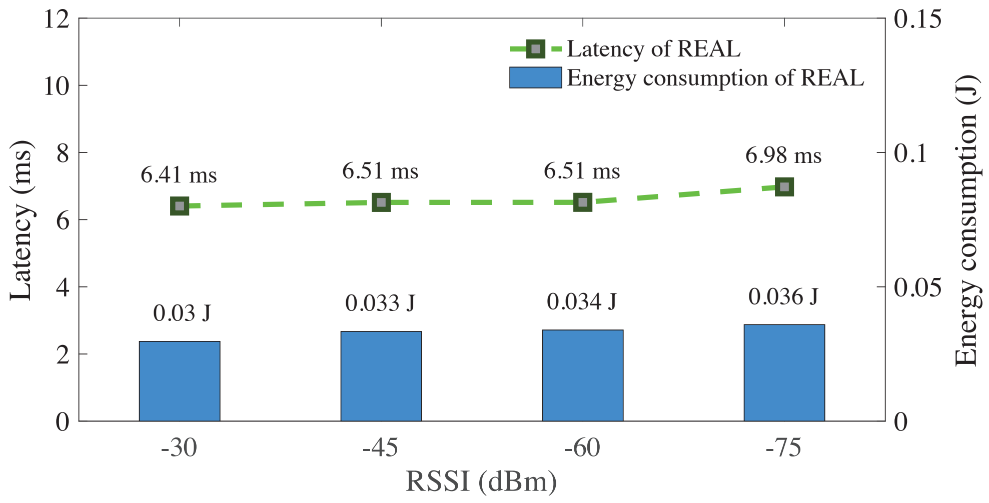


Figure 7.13: Latency and energy consumption of REAL at different RSSI levels and energy level 80%.

−30 dBm to −75 dBm, the latency and energy consumption rise by 9% and 21%, respectively at the battery energy level of 80%, which are shown in Fig. 7.13. This deviation is negligible as the learning part of REAL takes a significantly small portion of the overall latency and energy of the whole application.

Highlights: The latency and energy consumption of the encoding-based remote inference system due to offloading are both lower since the offloaded data size is much smaller than that of the remote inference only. *REAL edge-MAR reduces offloading latency and energy consumption due to changes in system states, showing significant improvement in QoS of an MAR application.* Apart from these, the experiment shows that *REAL action and decision handlers are affected negligibly by the degrading wireless link.*

7.4.5 Accuracy and Transmitted Data Size

The proposed REAL edge-MAR system should maintain a satisfactory accuracy level and reduce offloaded data size while optimizing the latency and energy consumption collectively. Fig. 7.14 presents the mean top-3 detection accuracy and mean offloaded data size of REAL edge-MAR at different RSSI and energy level 80%.

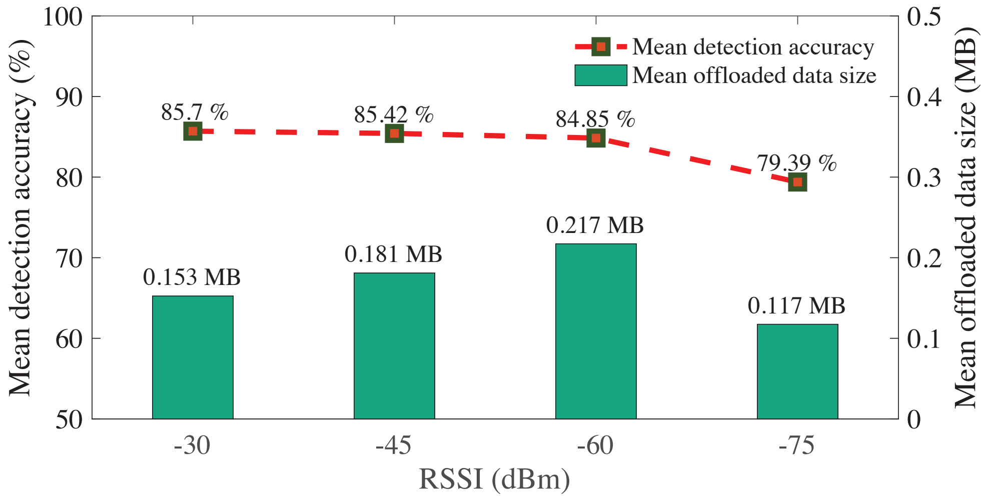


Figure 7.14: Mean detection accuracy and offloaded data size of REAL at different RSSI levels and energy level 80%.

Table 7.3: Mean detection accuracy (%) and offloaded data size (MB) by REAL during 100 episodes

Battery energy level (%)	**RSSI (dBm), accuracy (%), data size (MB)**							
	−30 **dBm**		−45 **dBm**		−60 **dBm**		−75 **dBm**	
	%	MB	%	MB	%	MB	%	MB
100	85.84	0.13	86.53	0.17	81.97	0	84.76	0.14
90	84.69	0.15	84.73	0.19	85.3	0.14	84.6	0.17
80	85.85	0.12	85.93	0.12	85.47	0.11	76.84	0
70	86.13	0.13	85.11	0.13	86.41	0.47	86.89	0.46
60	85.45	0	86.07	0.09	84.88	0.08	77.31	0
50	84.76	0.17	85.53	0.1	86.39	0.41	77.74	0
40	85.62	0.11	85.23	0.14	87.17	0.51	77.19	0
30	87.05	0.13	86.67	0.37	81.68	0	75.49	0.4
20	85.03	0.12	85.7	0.49	86.71	0.45	76.54	0
10	86.62	0.46	82.77	0	82.54	0	76.58	0

The average detection accuracy of REAL edge-MAR at all system states is 83.84%, which is 16% greater than that of the local inference only and close to remote inference systems.

Transmitted data size plays a critical role in utilizing the wireless network resources, e.g., bandwidth. Experimental data show that the REAL edge-MAR offloads a mean of 0.16 MB of data on average that is extremely smaller than that of remote inference, but slightly larger than that with encoding, which is a necessary trade-off to increase the QoS. A more detailed representation of the detection accuracy and offloaded data size at different system states is shown in Table 7.3. When REAL adopts local inference, the offloaded data size becomes zero, but at a slight cost of detection accuracy. The proposed system looks for an optimized trade-off to deal with challenges from varying system states.

Highlights: *REAL edge-MAR system maintains* 83.84% *accuracy and offloads* 0.16 *MB of data on average considering all the system states, which is reasonable with the optimization it brings to the system.*

7.4.6 Performance of REAL Edge-MAR at Different States

The performance of REAL edge-MAR is evaluated under various system states. From the experimental studies, it is observed that REAL takes suitable action decisions based on the system state information and continuously learns from past experiences to avoid taking wrong steps.

As mentioned in Section 7.2, the edge server calculates the reward based on (7.3) and (7.4) with normalized performance metric data. The weights are set for C_a, C_{off}, and $E_{B_{th}}$ as 0.2, 0.05, and 40% (experimental data show a drastic depth-of-discharge from 40%). Based on $E_{B_{th}}$, C_{l_1}, C_{l_2}, C_{e_1}, and C_{e_2} are set as 0.4, 0.35, 0.25, and 0.5 with a view to prolonging the battery life. After the completion of 100 training episodes with 200 runs, REAL edge-MAR provides around 87% better reward cost function value than the other baseline MAR systems. Fig. 7.15 shows the reward cost for the baseline systems and REAL edge-MAR at different RSSI and energy levels. REAL edge-MAR shows a lower reward cost value than the other systems at

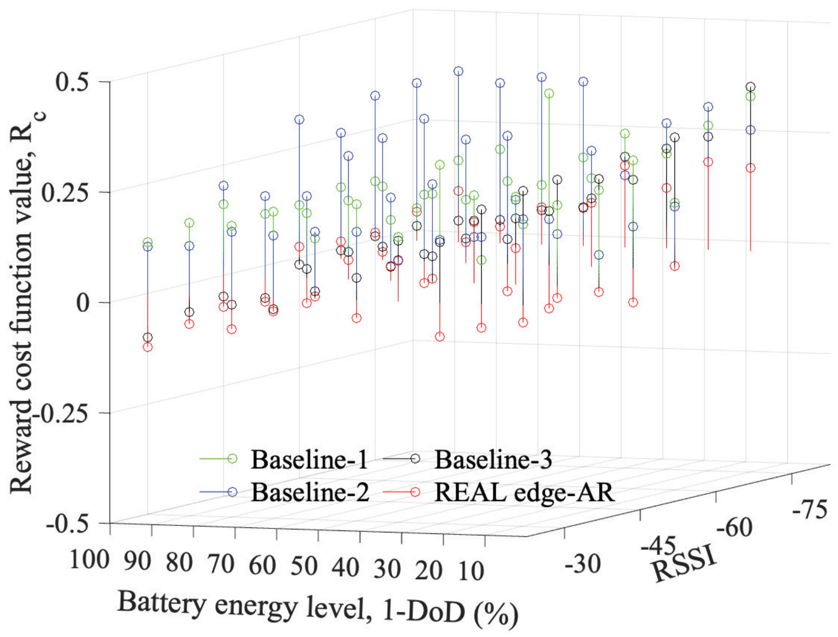


Figure 7.15: Reward cost function value at different energy and RSSI levels (the lower, the better).

Table 7.4: Mean rewards calculated by REAL during 100 episodes (before applying discount rate, γ)

Energy level (%)	**RSSI (dBm)**			
	-30	-45	-60	-75
100	1	1	0.74	0.84
90	1	1	0.82	0.76
80	1	1	1	–0.26
70	1	1	1	1
60	1	0.93	1	0.89
50	1	1	1	1
40	1	0.68	1	–0.17
30	1	1	1	1
20	1	1	1	1
10	1	1	1	1

almost every system states. This figure is explained in more detail by Table 7.4 with mean reward values. The mean reward value (before applying the discount rate, γ) of REAL after 100 episodes for the 40 system state combinations is 36.23, which asserts that REAL is able to increase the QoS at 90% system states.

Highlights: *After* 100 *training episodes, REAL can optimize the overall system parameters by lowering the reward cost function value by around* 87% *compared to other baseline systems. The mean reward value of REAL training before applying the discount rate is around* 90%*, which makes the entire optimization process lucrative.*

7.4.7 Performance of REAL Edge-MAR at Continuously Changing States

The performance of REAL is measured and evaluated in different system states. However, it is also necessary to observe its performance in continuously changing wireless link conditions and battery energy level conditions. We collect experimental data for thousands of continuous frames in changing states, where RSSI and remaining battery capacity values range from -30 dBm to -75 dBm and 100% to 20%, respectively. Fig. 7.16 shows some snippets of the overall latency (s), energy consumption (J), accuracy (%), and offloaded data size (MB) of REAL, along with actions taken at

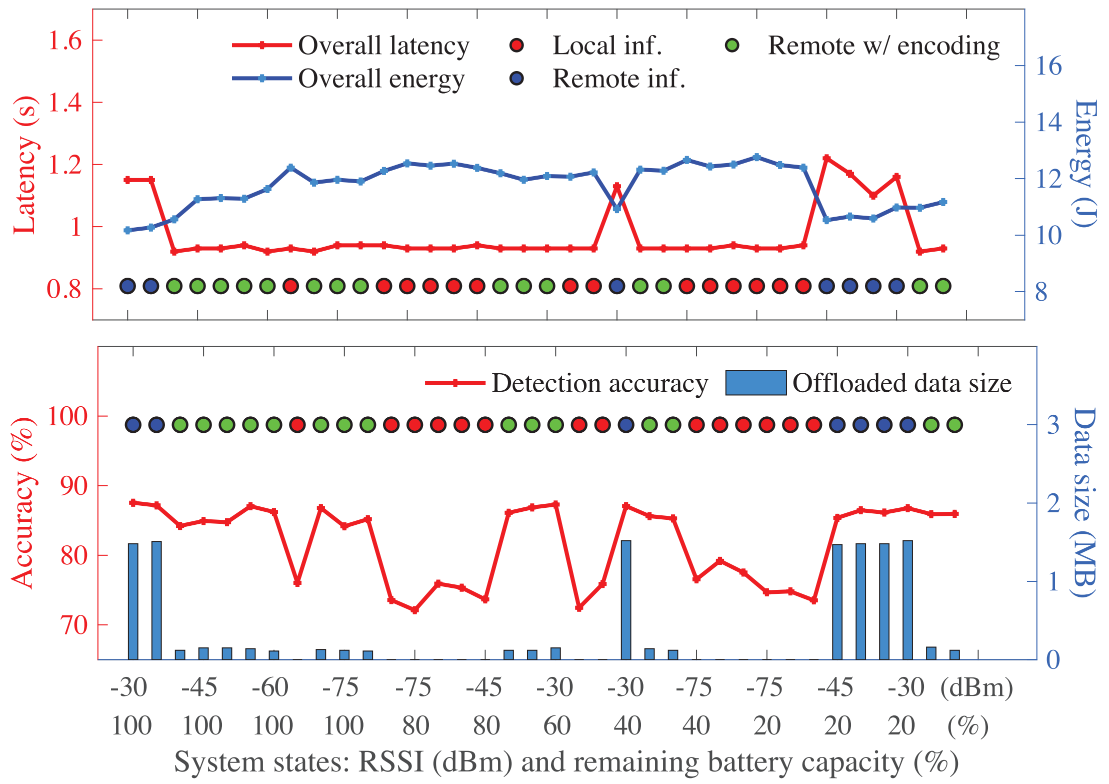


Figure 7.16: Overall latency, energy consumption, detection accuracy, and offloaded data size of REAL at continuously changing states.

different states continuously changing over a long period of time (the battery energy level cannot be decreased manually).

REAL takes different action decisions depending on the system states and previous reward experiences on optimizing the pre-set performance metrics according to the cost function defined in Section 7.2. The interesting observation here is, whenever the action taken by REAL causes higher energy, the latency and offloaded data size are reduced, keeping accuracy above at least 70% to ensure proper QoS, and vice-versa. In poor network conditions, REAL tries to reduce the offloaded data size. On the other hand, when the battery energy level or remaining battery capacity becomes low, REAL reduces energy consumption, keeping other parameters above an expected level to maintain a higher QoS. These characteristics of REAL makes it an effective solution in edge-MAR applications.

Highlights: *REAL takes advantage of a trade-off among the overall latency, energy, accuracy, and data size, depending on the changing system states.* The reward cost function defines the weights assigned to each of these four parameters, which dictates the operation of the learning algorithm. The way REAL takes actions based on the changing environments proves its effectiveness for such challenges in edge-based MAR applications.

7.4.8 Comparison of REAL with other state-of-the-art systems

Finally, the proposed smart decision framework "REAL" is compared with other state-of-the-art offloading systems in this subsection. The overall latency and energy consumption comparisons are shown in Fig. 7.17 and Fig. 7.18, respectively.

The state-of-the-art systems for offloading decisions used in this experiment for comparison are as follows:

- No-offloading scheme (NO offload): This scheme does not offload the data to the edge server and executes the inference locally only on the client device.
- Greedy offloading (Greedy offload): In this scheme, the data are always offloaded to the edge server for remote inference of MAR tasks.
- Q-learning method (Q-learning): This system uses the Q-learning approach where the temporal difference algorithm always tries to achieve the best reward by making offloading decisions (whether to offload or not to the edge server) [128].
- Dynamic reinforcement learning scheduling (DRLS): This is an offloading decision framework based on reinforcement learning that incorporates device-to-device and edge computing systems [129].
- Deep Reinforcement Learning-based offloading scheme for XR devices (DRLXR): This is a DRL-based offloading scheme for extended reality devices which

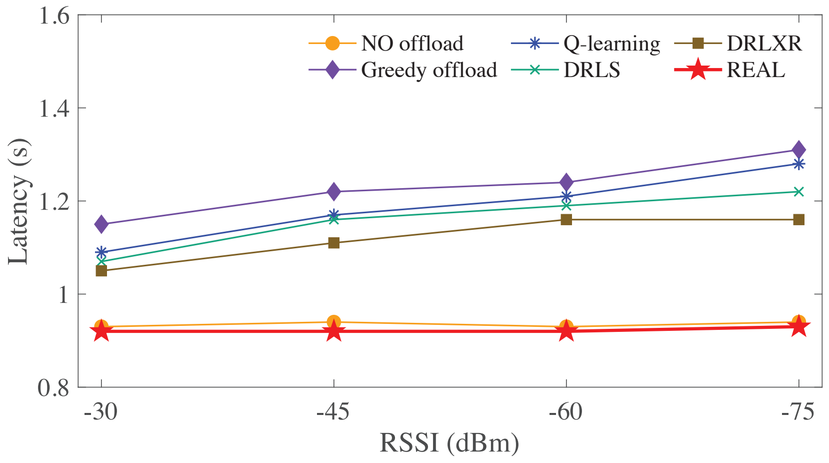


Figure 7.17: Comparison of overall latency of REAL with other systems.

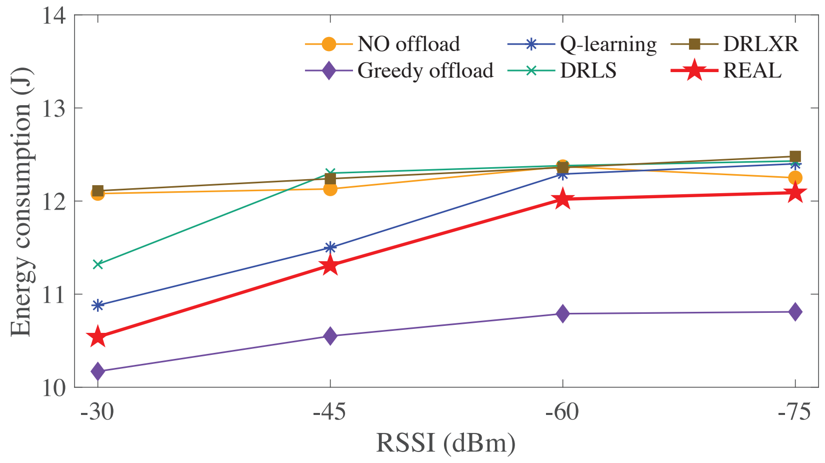


Figure 7.18: Comparison of overall energy consumption of REAL with other systems.

considers complete or partial offloading to the edge server [9].

These systems are chosen for comparison as most of them consider reinforcement learning as the tool to make offloading decisions. They are implemented using the physical testbed and collect data for comparison purposes. In varying wireless link conditions, these methods do not suit well in terms of latency and energy consumption. Moreover, the greedy offloading scheme may exhibit low energy consumption but at a cost of high latency. The overall mean latency and energy consumption of REAL are 18% and 7.5% lower than that of the other systems presented in this comparison, respectively. The proposed decision framework “REAL” outperforms all the other systems in varying wireless link conditions by trading off among latency, energy consumption, service accuracy, and offloaded data size.

Highlights: REAL considers the varying wireless link conditions and the remaining battery capacity of a mobile device for MAR applications, whereas the other state-of-the-art reinforcement learning-based systems only focus on the offloading decisions with the purpose of reducing the overall latency and energy consumption. *REAL trades-off among end-to-end latency, energy consumption, service accuracy, and offloaded data size, which brings out the most effective QoS in an MAR application in changing environments.*

CHAPTER 8: Conclusion

8.1 Completed Work

This dissertation advocates the necessity of providing an energy-efficient infrastructure for mobile AI applications to enhance their performance, considering edge computing.

First, this dissertation presented a comprehensive study of mobile AI applications with different processing sources and AI models. Overcoming the challenges with measurement, experiments were conducted to assess the performance of different AI models, processing sources, and devices. The measurement work shows that the latency, energy consumption, and memory usage vary based on DNN models and processing sources. Mobile AI systems' performance is substantially improved using quantized models than floating-point models in terms of latency and energy. Another important finding is that the storage space occupied by DNN models influences the memory and energy consumed during inference almost linearly. Additionally, non-vision applications follow a different trend of latency and energy consumption than vision-based AI since their input processing techniques differ from vision applications. Every AI application has an initiation delay caused by accessing various hardware components of mobile devices, which varies for different models and configurations. Moreover, the latency, memory, AI model, and device configuration impact the total energy consumption for a complete application cycle, albeit at different correlations. This non-linear correlation in a non-parametric model led to the proposed predictive energy model, EPAM, based on Gaussian process regression. In this research, EPAM was trained and validated with the vast dataset obtained from the experiment. The evaluation of EPAM shows high accuracy with an overall RMSE of 0.075 (3.06%).

Developers can use EPAM to predict the energy consumption of their mobile AI applications without measuring the energy externally to improve the comprehensive user experience. To summarize, this novel predictive energy model, EPAM, will help the mobile AI research community design energy-improved applications considering all the control factors and parameters that can reduce energy requirements to enable better service for smartphones, wearable devices, and autonomous vehicles.

Second, a novel modeling framework is presented for performance analysis of XR applications in edge-assisted wireless networks. The proposed framework consists of analytical methods to evaluate the performance of individual segments of an XR pipeline in terms of end-to-end latency, energy consumption, and AoI. In wireless networks, the mobility of XR devices causes unique transmission and HO delays. Moreover, information from heterogeneous sensors and devices sent to the XR device produces an additional load on the buffer, introducing delays in the overall latency and an increase in energy consumption. In addition, external sensors and devices generate information at their own frequencies, which may cause improper arrival of information packets in the XR pipeline. The proposed model becomes comprehensive by taking all these details into account, which have not been considered in existing work. Consequently, the proposed analytical model was validated against ground truth and compared with state-of-the-art models. The evaluation shows that the proposed performance analysis modeling framework for XR applications effectively captures and incorporates the important determining factors affecting the end-to-end latency, energy consumption, and AoI, and thus performs better than the compared models with high accuracy.

Third, in this research, a novel service aggregation system was proposed for time-sensitive CAV applications based on predictive AoI. The initial study indicated that due to the low coverage areas of connected roadside sensors and nearby vehicles to a CAV, the high mobility of the CAV causes severe degradation in AoI, which in turn makes service aggregation even more challenging. To address this challenge, a service aggregation system was proposed that uses AoI prediction at a specific interval and clustering information source nodes based on the predicted AoI. The periodic prediction and clustering of source nodes help reduce the computational load and latency. Simulation results showed that the proposed system is capable of predicting the AoI with high accuracy and providing high DSSR while maintaining the AoI threshold for low-latency CAV applications. Lastly, the performance comparison of the proposed system with other data sequencing methods showed the superiority of the proposed service aggregation system in high-mobility CAV scenarios.

Fourth, a detailed, comprehensive experimental study of network and energy-resource utilization by an Edge-MAR with H.264 in different wireless network conditions was presented in this research for a variety of mobile devices. The measurement study showed that the use of H.264 in Edge-MAR can substantially reduce latency, but at the cost of slightly increased energy consumption – especially in worse wireless network conditions. It was observed that with the increase in CPU frequency and frame size, the overall transmission and encoding latency and energy consumption varies to a great extent, but at some specific frequencies and frame sizes, the variations are different due to smartphones' efficiency issues. The study showed the necessity of trade-offs among Edge-MAR parameters to achieve desired outcomes. Lastly, regression-based models were proposed to design Edge-MAR systems with H.264 encoding. Any MAR system involving video transmission can leverage the benefits of these proposed models. In short, the findings from this research will provide great insights into further designs of latency- and energy-aware Edge-MAR pipelines with H.264.

In the end, this dissertation proposed a novel deep reinforcement learning-based edge-MAR system – REAL, which makes dynamic and smart decisions to deal with the varying wireless link quality and mobile devices' battery energy levels. Since the change in the battery energy level of a mobile device is guided by the hardware configurations, camera parameters, computation resource allocation, and battery health, it represents the overall computing system of the device. With the changes under these system states, the key performance metrics of an MAR system also experience variations. As a result, the proposed REAL edge-MAR system ensures increased QoS of an MAR application. The approach to the stochastic nature of the state transition problem, followed by discrete realistic actions for processing and offloading frames with an off-policy soft actor-critic online learning method, sets the work apart from the state-of-the-art research. Through extensive experimental study with a physical testbed, the algorithms of REAL was implemented on different mobile devices and an edge server. The experiments showed that the proposed REAL edge-MAR system outperforms all the baseline MAR systems by balancing the end-to-end latency, energy consumption, mean service accuracy, and offloaded data size of mobile devices without introducing additional load to the system. The training of REAL was validated with reward calculations using a reward cost function under varying combinations of states. REAL was tested in continuously changing system states in terms of actions taken, latency, energy, service accuracy, and offloaded data size. Finally, the proposed system was compared to other state-of-the-art reinforcement learning-based offloading schemes for edge-MAR systems, where REAL outperformed all the schemes in different wireless link conditions, since the other systems do not consider the changing wireless network and battery energy levels. To summarize, REAL edge-MAR increases the overall QoS of an MAR system under dynamic system states by making smart decisions using online deep reinforcement learning.

8.2 Future Work

This Ph.D. research opens up many theoretical and experimental research possibilities in providing energy-efficient wireless infrastructures for sustainable AI applications. This dissertation directs but does not limit future efforts to the following research topics:

- **Performance enhancement of edge-assisted AI applications in heterogeneous wireless networks:**

  The performance analysis modeling framework presented in this dissertation shows the importance of AI applications' critical performance metrics (latency, energy consumption, and AoI) to achieve superior QoS. However, there is always a trade-off going on among these performance metrics. Depending on the application requirements, these metrics need to be optimized. For example, a highly mobile user may encounter larger latency, which can impede the QoS. On the contrary, devices with smaller battery capacity may need to save energy for a longer period by reducing the XR application's energy consumption. Moreover, an autonomous vehicle will need the highest possible accuracy to ensure safety on the road. Additionally, the heterogeneity of wireless networks and devices poses additional challenges in the dissemination of environmental information. This dissertation is currently based on the fundamental objective of performance enhancement for edge-assisted AI applications, which has the potential to pursue this future direction.

- **Energy-efficient Generative AI application for mobile devices:**

  Recent developments in Generative AI (Gen-AI) have made related applications quite popular all around the world. Starting from GPT models to generative image and video content are being used heavily by users. However, these applications consume extremely high computational resources, which in turn increases the energy consumption. To improve the QoS of these Gen-AI applications, the latency needs to be reduced to some point, as well. All these issues make this research direction interesting. Right now, mobile devices with state-of-the-art hardware configurations are incapable of providing such QoS. The research question is: *how can edge computing help Gen-AI applications running on mobile devices to provide better QoS?* This future research direction is well-aligned with the research problem this dissertation has approached to solve.

8.3 Published and Submitted Work

The following list is a summary of my publications and submitted works from this dissertation.

8.3.1 Journal Papers

1. **Anik Mallik**, Dawei Chen, Kyungtae Han, Jiang Xie, and Zhu Han, *"REAL: Where mobile augmented reality meets the smart edge,"* under major revision for IEEE Transactions on Mobile Computing, 2024.

2. **Anik Mallik**, Haoxin Wang, Dawei Chen, Kyungtae Han, and Jiang Xie, *"EPAM 2.0: Deep reinforcement learning-based energy prediction for mobile AI vision and non-vision applications with large language models,"* submitted to IEEE Transactions on Mobile Computing, 2024.

3. Xiaolong Tu, **Anik Mallik**, Dawei Chen, Kyungtae Han, Onur Altintas, Haoxin Wang, and Jiang Xie, *"SDNN: Designing sustainable deep neural networks via datasets, predictors, and neural architecture search,"* submitted to IEEE Transactions on Mobile Computing, 2024.

4. Xiaolong Tu, **Anik Mallik**, Muhana Magboul Ali Muslam, Haoxin Wang, and Jiang Xie, *"AIEnergy: An energy benchmark for AI-empowered mobile and IoT*

devices," submitted to IEEE Internet of Things Journal, 2024.

5. **Anik Mallik**, Dawei Chen, Kyungtae Han, Jiang Xie, and Zhu Han, *"You can have it all: Performance analysis and enhancement of extended reality in heterogeneous wireless network and device environments,"* in preparation to submit to IEEE Transactions on Mobile Computing, 2024.

8.3.2 Conference Papers

1. **Anik Mallik**, Jiang Xie, and Zhu Han, *"A performance analysis modeling framework for edge computing-enabled extended reality applications in high-mobility wireless networks,"* in Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), 2024. **(acceptance rate 21.9%)**

2. **Anik Mallik**, Dawei Chen, Kyungtae Han, Jiang Xie, and Zhu Han, *"Unleashing the true power of Age-of-Information: Service aggregation in connected and autonomous vehicles,"* in Proceedings of IEEE International Conference on Communications (ICC), 2024.

3. Xiaolong Tu, **Anik Mallik**, Haoxin Wang, and Jiang Xie, *"DeepEn2023: Energy datasets for edge artificial intelligence,"* in Proceedings of NeurIPS 2023 Workshop: Tackling Climate Change with Machine Learning, 2023.

4. Xiaolong Tu, **Anik Mallik**, Dawei Chen, Kyungtae Han, Onur Altintas, Haoxin Wang, and Jiang Xie, *"Unveiling energy efficiency in deep learning: Measurement, prediction, and scoring across edge devices,"* in Proceedings of ACM/IEEE Symposium on Edge Computing (SEC), 2023. **(acceptance rate 25%)**

5. **Anik Mallik**, Hoaxin Wang, Jiang Xie, Dawei Chen, and Kyungtae Han, *"EPAM: A predictive energy model for mobile AI,"* in Proceedings of IEEE International Conference on Communications (ICC), 2023.

6. **Anik Mallik** and Jiang Xie, *"H.264 video encoding-based edge-assisted mobile AR systems: Network and energy Issues,"* in Proceedings of IEEE International Conference on Communications (ICC), 2022.

REFERENCES

[1] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Honolulu, HI), pp. 756–764, 2018.

[2] H. Wang, B. Kim, J. Xie, and Z. Han, "LEAF + AIO: Edge-assisted energy-aware object detection for mobile augmented reality," *IEEE Transactions on Mobile Computing*, vol. 22, pp. 5933–5948, October 2023.

[3] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, Nevada, USA), pp. 4820–4828, 2016.

[4] L. N. Huynh, R. K. Balan, and Y. Lee, "DeepSense: A GPU-based deep convolutional neural network framework on commodity mobile devices," in *Proceedings of ACM Workshop on Wearable Systems and Applications*, pp. 25–30, 2016.

[5] M.-Y. Lai, C.-Y. Sung, J.-K. Lee, and M.-Y. Hung, "Enabling Android NNAPI flow for TVM runtime," in *Proceedings of ACM ICPP: Workshops*, pp. 1–8, 2020.

[6] I. F. Akyildiz and H. Guo, "Wireless extended reality (XR): Challenges and new research directions," *ITU Journal on Future and Evolving Technologies*, vol. 3, no. 2, pp. 1–15, 2022.

[7] H. Wang, B. Kim, J. Xie, and Z. Han, "Energy drain of the object detection processing pipeline for mobile devices: analysis and implications," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 41–60, March 2021.

[8] T. Theodoropoulos, A. Makris, A. Boudi, T. Taleb, U. Herzog, L. Rosa, L. Cordeiro, K. Tserpes, E. Spatafora, A. Romussi, E. Zschau, M. Kamarianakis, A. Protopsaltis, G. Papagiannakis, and P. Dazzi, "Cloud-based XR services: A survey on relevant challenges and enabling technologies," *Journal of Networking and Network Applications*, vol. 2, pp. 1–22, February 2022.

[9] B. Trinh and G.-M. Muntean, "A deep reinforcement learning-based offloading scheme for multi-access edge computing-supported eXtended reality systems," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1254–1264, January 2023.

[10] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[11] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, October 2018.

[12] X. Ran, H. Chen, Z. Liu, and J. Chen, "Delivering deep learning to mobile devices via offloading," in *Proceedings of ACM Workshop on Virtual Reality and Augmented Reality Network*, (Los Angeles, CA), pp. 42–47, 2017.

[13] I. F. Akyildiz and H. Guo, "Wireless communication research challenges for extended reality (XR)," *ITU Journal on Future and Evolving Technologies*, vol. 3, pp. 1–15, September 2022.

[14] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2018.

[15] X. Ma, J. Zhang, X. Yin, and K. S. Trivedi, "Design and analysis of a robust broadcast scheme for VANET safety-related services," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 46–61, 2011.

[16] F. Lyu, H. Zhu, H. Zhou, W. Xu, N. Zhang, M. Li, and X. Shen, "SS-MAC: A novel time slot-sharing MAC for safety messages broadcasting in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3586–3597, 2017.

[17] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[18] C. Luo, X. He, J. Zhan, L. Wang, W. Gao, and J. Dai, "Comparison and benchmarking of AI models and frameworks on mobile devices," *arXiv preprint arXiv:2005.05085*, 2020.

[19] A. Mallik and J. Xie, "H.264 video encoding-based edge-assisted mobile AR systems: Network and energy issues," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Seoul, South Korea), pp. 1046–1051, 2022.

[20] X. Tu, A. Mallik, D. Chen, K. Han, O. Altintas, H. Wang, and J. Xie, "Unveiling energy efficiency in deep learning: Measurement, prediction, and scoring across edge devices," in *Proceedings of ACM/IEEE Symposium on Edge Computing (SEC)*, (Wilmington, DE), pp. 80–93, 2023.

[21] X. Tu, A. Mallik, H. Wang, and J. Xie, "Deepen2023: Energy datasets for edge artificial intelligence," in *Proceedings of NeurIPS 2023 Workshop: Tackling Climate Change with Machine Learning*, (New Orleans, LA), 2023.

[22] A. Mallik, H. Wang, J. Xie, D. Chen, and K. Han, "EPAM: A Predictive Energy Model for Mobile AI," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Rome, Italy), pp. 954–959, 2023.

[23] M. Seeger, "Gaussian processes for machine learning," *International Journal of Neural Systems*, vol. 14, no. 02, pp. 69–106, 2004.

[24] A. Mallik, J. Xie, and Z. Han, "A Performance Analysis Modeling Framework for Extended Reality Applications in Edge-Assisted Wireless Networks," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, (Jersey City, NJ), pp. 1–12, 2024.

[25] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled Metaverse," *IEEE Wireless Communications Letters*, vol. 11, pp. 1733–1737, August 2022.

[26] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, pp. 5225–5240, August 2018.

[27] M. M. I. Rajib and A. Nasipuri, "Predictive retransmissions for intermittently connected sensor networks with transmission diversity," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, pp. 1–25, September 2017.

[28] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proceedings of IEEE International Conference on Edge Computing (EDGE)*, (San Francisco, CA), pp. 66–73, 2018.

[29] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury, "Frugal following: Power thrifty object detection and tracking for mobile augmented reality," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, (New York, NY), pp. 96–109, 2019.

[30] Y. Leng, C.-C. Chen, Q. Sun, J. Huang, and Y. Zhu, "Energy-efficient video processing for virtual reality," in *Proceedings of IEEE/ACM International Symposium on Computer Architecture (ISCA)*, (Phoenix, AZ), pp. 91–103, 2019.

[31] C. Xu, Q. Xu, J. Wang, K. Wu, K. Lu, and C. Qiao, "AoI-centric task scheduling for autonomous driving systems," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (London, United Kingdom), pp. 1019–1028, 2022.

[32] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Age-of-Information for computation-intensive messages in mobile edge computing," in *Proceedings of IEEE International Conference on Wireless Communications and Signal Processing (WCSP)*, (Xi'an, China), pp. 1–6, 2019.

[33] S. Lee, K. Sriram, K. Kim, Y. H. Kim, and N. Golmie, "Vertical handoff decision algorithms for providing optimized performance in heterogeneous wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 865–881, February 2008.

[34] Q. Zhang, N. Cheng, R. Sun, and D. Zhang, "AoI-Oriented context-aware priority design and vehicle scheduling strategy in vehicular networks," in *Proceedings of IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 233–238, 2022.

[35] M. Ghoshal, I. Khan, Z. J. Kong, P. Dinh, J. Meng, Y. C. Hu, and D. Koutsonikolas, "Performance of cellular networks on the wheels," in *Proceedings of ACM Internet Measurement Conference*, pp. 678–695, 2023.

[36] F. H. Administration, "Traffic control systems handbook: Chapter 6 detectors–FHWA office of operations," 2018.

[37] "Vehicle-to-vehicle communication." https://www.nhtsa.gov/technology-innovation/vehicle-vehicle-communication.

[38] T. Kim, A. G. Hobeika, H. Jung, *et al.*, "Area coverage provided by vehicle to vehicle communication in an urban network," tech. rep., Mid-Atlantic Universities Transportation Center, 2014.

[39] A. Mallik, D. Chen, K. Han, J. Xie, and Z. Han, "Unleashing the true power of Age-of-Information: Service aggregation in connected and autonomous vehicles," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Denver, CO), pp. 1–6, 2024.

[40] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 4, pp. 1–43, 2008.

[41] S. Huang, J. Xie, and M. Muslam, "A cloud computing based deep compression framework for UHD video delivery," *IEEE Transactions on Cloud Computing*, 2022.

[42] C. Vidal, O. Gross, R. Gu, P. Kollmeyer, and A. Emadi, "xEV li-ion battery low-temperature effects," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4560–4572, May 2019.

[43] X. Chen, A. Jindal, N. Ding, Y. C. Hu, M. Gupta, and R. Vannithamby, "Smartphone background activities in the wild: Origin, energy drain, and optimization," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, (Paris, France), pp. 40–52, 2015.

[44] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "AI benchmark: All about deep learning on smartphones in 2019," in *Proceedings of IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, (Seoul, South Korea), pp. 3617–3635, 2019.

[45] J. Liu, J. Liu, W. Du, and D. Li, "Performance analysis and characterization of training deep learning models on mobile device," in *Proceedings of IEEE ICPADS*, pp. 506–515, 2019.

[46] R. Yazdani, A. Segura, J.-M. Arnau, and A. Gonzalez, "An ultra low-power hardware accelerator for automatic speech recognition," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, 2016.

[47] S. Kim, J. Lee, S. Kang, J. Lee, and H.-J. Yoo, "A power-efficient cnn accelerator with similar feature skipping for face recognition in mobile devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 4, pp. 1181–1193, 2020.

[48] D. Shin, J. Lee, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An energy-efficient deep-learning processor with heterogeneous multi-core architecture," *IEEE Micro*, vol. 38, no. 5, pp. 85–93, 2018.

[49] Y.-M. Chang, C.-Y. Sung, Y.-C. Sheu, M.-S. Yu, M.-Y. Hsu, and J.-K. Lee, "Support NNEF execution model for NNAPI," *The Journal of Supercomputing*, vol. 77, no. 9, pp. 10065–10096, 2021.

[50] J. Lee, S. Kang, J. Lee, D. Shin, D. Han, and H.-J. Yoo, "The hardware and algorithm co-design for energy-efficient dnn processor on edge/mobile devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 10, pp. 3458–3470, 2020.

[51] M. Motamedi, D. Fong, and S. Ghiasi, "Machine intelligence on resource-constrained iot devices: The case of thread granularity optimization for cnn inference," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–19, 2017.

[52] Z. Lu, S. Rallapalli, K. Chan, and T. La Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *Proceedings of ACM International Conference on Multimedia*, pp. 1663–1671, 2017.

[53] B. Sun, D. Liu, L. Yu, J. Li, H. Liu, W. Zhang, and T. Torng, "MRAM co-designed processing-in-memory CNN accelerator for mobile and IoT applications," *arXiv preprint arXiv:1811.12179*, 2018.

[54] Z. Xu, F. Yu, Z. Qin, C. Liu, and X. Chen, "DiReCtX: Dynamic resource-aware CNN reconfiguration framework for real-time mobile applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 246–259, 2020.

[55] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized CNN: A unified approach to accelerate and compress convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4730–4743, 2017.

[56] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[57] J. Johnson, "Rethinking floating point for deep learning," *arXiv preprint arXiv:1811.01721*, 2018.

[58] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[59] Y. Boo, S. Shin, and W. Sung, "Quantized neural networks: Characterization and holistic optimization," in *Proceedings of IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, 2020.

[60] R. Goyal, J. Vanschoren, V. Van Acht, and S. Nijssen, "Fixed-point quantization of convolutional neural networks for quantized inference on embedded platforms," *arXiv preprint arXiv:2102.02147*, 2021.

[61] R. Xie, X. Jia, L. Wang, and K. Wu, "Energy efficiency enhancement for CNN-based deep mobile sensing," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 161–167, 2019.

[62] T. Zhao, Y. Xie, Y. Wang, J. Cheng, X. Guo, B. Hu, and Y. Chen, "A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities," *Proceedings of IEEE*, vol. 110, no. 3, pp. 334–354, 2022.

[63] F. N. Iandola, A. E. Shaw, R. Krishna, and K. W. Keutzer, "SqueezeBERT: What can computer vision teach NLP about efficient neural networks?," in *Proceedings of Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, 2020.

[64] T. Tambe, C. Hooper, L. Pentecost, T. Jia, E.-Y. Yang, M. Donato, V. Sanh, P. Whatmough, A. M. Rush, D. Brooks, *et al.*, "EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO-54)*, pp. 830–844, 2021.

[65] S. Zheng, P. Ouyang, D. Song, X. Li, L. Liu, S. Wei, and S. Yin, "An ultra-low power binarized convolutional neural network-based speech recognition processor with on-chip self-learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 12, pp. 4648–4661, 2019.

[66] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, *et al.*, "Personalized speech recognition on mobile devices," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5955–5959, 2016.

[67] B. Delaney, N. Jayant, and T. Simunic, "Energy-aware distributed speech recognition for wireless mobile devices," *IEEE Design & Test of Computers*, vol. 22, no. 1, pp. 39–49, 2005.

[68] M. Liubogoshchev, K. Ragimova, A. Lyakhov, S. Tang, and E. Khorov, "Adaptive cloud-based extended reality: Modeling and optimization," *IEEE Access*, vol. 9, pp. 35287–35299, 2021.

[69] M. Huzaifa, R. Desai, S. Grayson, X. Jiang, Y. Jing, J. Lee, F. Lu, Y. Pang, J. Ravichandran, F. Sinclair, B. Tian, H. Yuan, J. Zhang, and S. V. Adve, "ILLIXR: Enabling end-to-end extended reality research," in *Proceedings of IEEE International Symposium on Workload Characterization (IISWC)*, (Storrs, CT), pp. 24–38, 2021.

[70] P. Caricato, L. Colizzi, M. G. Gnoni, A. Grieco, A. Guerrieri, and A. Lanzilotto, "Augmented reality applications in manufacturing: A multi-criteria decision model for performance analysis," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 754–759, 2014.

[71] S. Sendari, A. Firmansah, and Aripriharta, "Performance analysis of augmented reality based on Vuforia using 3D marker detection," in *Proceedings of IEEE International Conference on Vocational Education and Training (ICOVET)*, (Malang, Indonesia), pp. 294–298, 2020.

[72] D. A. Reed, K. A. Shields, W. H. Scullin, L. F. Tavera, and C. L. Elford, "Virtual reality and parallel systems performance analysis," *IEEE Computer*, vol. 28, pp. 57–67, November 1995.

[73] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski, "Latency and throughput characterization of convolutional neural networks for mobile computer vision," in *Proceedings of ACM Multimedia Systems Conference*, pp. 204–215, 2018.

[74] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE*, vol. 107, pp. 1717–1737, August 2019.

[75] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, A. de la Oliva, and A. Azcorra, "Modeling mobile edge computing deployments for low latency multimedia services," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 464–474, 2019.

[76] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proceedings of IEEE European Conference on Networks and Communications (EuCNC)*, pp. 1–6, 2017.

[77] N. T. Hoa, L. V. Huy, B. D. Son, N. C. Luong, and D. Niyato, "Dynamic offloading for edge computing-assisted Metaverse systems," *IEEE Communications Letters*, vol. 27, pp. 1749–1753, July 2023.

[78] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, (San Francisco, CA), pp. 1–6, 2017.

[79] T. Braud, Z. Pengyuan, J. Kangasharju, and H. Pan, "Multipath computation offloading for mobile augmented reality," in *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom)*, (Austin, TX), pp. 1–10, 2020.

[80] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz, "MARVEL: Enabling mobile augmented reality with low energy and low latency," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, (Shenzhen, China), pp. 292–304, 2018.

[81] J. Cao, X. Zhu, S. Sun, Z. Wei, Y. Jiang, J. Wang, and V. K. Lau, "Toward industrial Metaverse: Age of Information, latency and reliability of short-packet transmission in 6G," *IEEE Wireless Communications*, vol. 30, pp. 40–47, April 2023.

[82] F. Chiariotti, O. Vikhrova, B. Soret, and P. Popovski, "Peak Age of Information distribution for edge computing with wireless links," *IEEE Transactions on Communications*, vol. 69, pp. 3176–3191, May 2021.

[83] M. Chen, Y. Xiao, Q. Li, and K.-C. Chen, "Minimizing Age-of-Information for fog computing-supported vehicular networks with deep Q-learning," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Dublin, Ireland), pp. 1–6, 2020.

[84] A. Muhammad, I. Sorkhoh, M. Samir, D. Ebrahimi, and C. Assi, "Minimizing Age of Information in multiaccess-edge-computing-assisted IoT networks," *IEEE Internet of Things Journal*, vol. 9, pp. 13052–13066, August 2021.

[85] X. Zhang, H. Wang, W. Feng, and S. Lin, "Vehicle environment awareness based messages transmission frequency optimization in C-V2X," *IEEE Wireless Communications Letters*, vol. 12, pp. 1116–1119, July 2023.

[86] E. F. Nakamura, A. A. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, pp. 9–es, 2007.

[87] J. Yuea, W. Zhang, W. Xiao, D. Tang, and J. Tang, "Energy efficient and balanced cluster-based data aggregation algorithm for wireless sensor networks," *Procedia Engineering*, vol. 29, pp. 2009–2015, 2012.

[88] A. M. Srivatsa and J. Xie, "A performance study of mobile handoff delay in IEEE 802.11-based wireless mesh networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 2485–2489, 2008.

[89] W. Nasrin and J. Xie, "SharedMEC: Sharing clouds to support user mobility in mobile edge computing," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Kansas City, MO), pp. 1–6, 2018.

[90] Q. Qi, L. Zhang, J. Wang, H. Sun, Z. Zhuang, J. Liao, and F. R. Yu, "Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13861–13874, 2020.

[91] Y. Ni, L. Cai, and Y. Bo, "Vehicular beacon broadcast scheduling based on Age of Information (AoI)," *China Communications*, vol. 15, no. 7, pp. 67–76, 2018.

[92] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof," in *Proceedings of ACM European Conference on Computer Systems*, pp. 29–42, 2012.

[93] A. Jindal and Y. C. Hu, "Experience: Developing a usable battery drain testing and diagnostic tool for the mobile industry," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 804–815, 2021.

[94] Q. Cao, A. Balasubramanian, and N. Balasubramanian, "Towards accurate and reliable energy measurement of NLP models," in *Proceedings of Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, 2020.

[95] T. Ullah, A. H. Siraj, U. M. Andrabi, and A. Nazarov, "Approximating and predicting energy consumption of portable devices," in *Proceedings of IEEE International Conference on Information Technology and Nanotechnology (ITNT)*, pp. 1–7, 2022.

[96] J. M. Vatjus-Anttila, T. Koskela, and S. Hickey, "Power consumption model of a mobile GPU based on rendering complexity," in *Proceedings of IEEE International Conference on Next Generation Mobile Apps, Services and Technologies*, (Prague, Czech Republic), pp. 210–215, 2013.

[97] "Trepn power profiler by Qualcomm." https://www.apkmirror.com/apk/qualcomm-innovation-center-inc/trepn-profiler/.

[98] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, (Scottsdale, AZ), pp. 105–114, 2010.

[99] H. Trinh, P. Calyam, D. Chemodanov, S. Yao, Q. Lei, F. Gao, and K. Palaniappan, "Energy-aware mobile edge computing and routing for low-latency visual data processing," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2562–2577, 2018.

[100] S. Aggarwal, M. Ghoshal, P. Banerjee, D. Koutsonikolas, and J. Widmer, "802.11 ad in smartphones: energy efficiency, spatial reuse, and impact on applications," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Vancouver, British Columbia, Canada), pp. 1–10, 2021.

[101] K. Apicharttrisorn, B. Balasubramaniany, J. Chen, R. Sivarajz, Y.-Z. Tsai, R. Janay, S. Krishnamurthy, T. Trany, and Y. Zhouy, "Characterization of multi-user augmented reality over cellular networks," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, 2020.

[102] H. Wang and J. Xie, "User preference based energy-aware mobile AR system with edge computing," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Toronto, ON, Canada), pp. 1379–1388, 2020.

[103] H. Wang, B. Kim, J. Xie, and Z. Han, "How is energy consumed in smartphone deep learning apps? Executing locally vs. remotely," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, (Waikoloa, HI), pp. 1–6, 2019.

[104] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Toronto, ON, Canada), pp. 1–10, 2020.

[105] L. Stäcker, J. Fei, P. Heidenreich, F. Bonarens, J. Rambach, D. Stricker, and C. Stiller, "Deployment of deep neural networks for object detection on edge AI devices with runtime optimization," in *Proceedings of IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, (Montreal, BC, Canada), pp. 1015–1022, 2021.

[106] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1529–1541, July-Sept. 2019.

[107] N. Didar and M. Brocanelli, "eAR: An edge-assisted and energy-efficient mobile augmented reality framework," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3898–3909, July 2023.

[108] Z. Ning, J. Huang, X. Wang, J. J. Rodrigues, and L. Guo, "Mobile edge computing-enabled Internet of Vehicles: Toward energy-efficient scheduling," *IEEE Network*, vol. 33, no. 5, pp. 198–205, Sept.-Oct. 2019.

[109] X. Li, Y. Tian, F. Zhang, S. Quan, and Y. Xu, "Object detection in the context of mobile augmented reality," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, (Porto de Galinhas, Brazil), pp. 156–163, 2020.

[110] L. Tobías, A. Ducournau, F. Rousseau, G. Mercier, and R. Fablet, "Convolutional neural networks for object recognition on mobile devices: A case study," in *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*, (Cancun, Mexico), pp. 3530–3535, 2016.

[111] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, (Seoul, South Korea), pp. 155–168, 2015.

[112] U. Drolia, K. Guo, and P. Narasimhan, "Precog: Prefetching for image recognition applications at the edge," in *Proceedings of ACM/IEEE Symposium on Edge Computing (SEC)*, (San Jose, CA), pp. 1–13, 2017.

[113] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Honolulu, HI), pp. 1421–1429, 2018.

[114] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Paris, France), pp. 1459–1467, 2019.

[115] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Paris, France), pp. 1270–1278, 2019.

[116] W. Zhang, S. Lin, F. H. Bijarbooneh, H. F. Cheng, and P. Hui, "CloudAR: A cloud-based framework for mobile augmented reality," in *Proceedings of Thematic Workshops of ACM Multimedia*, (Mountain View, CA), pp. 194–200, 2017.

[117] E. Meskar and B. Liang, "Fair multi-resource allocation in mobile edge computing with multiple access points," in *Proceedings of ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc)*, pp. 11–20, 2020.

[118] T. H. L. Dinh, M. Kaneko, K. Wakao, K. Kawamura, T. Moriyama, H. Abeysekera, and Y. Takatori, "Distributed user-to-multiple access points association through deep learning for beyond 5G," *Computer Networks*, vol. 197, p. 108258, 2021.

[119] Q. Li, J. Zhao, and Y. Gong, "Computation offloading and resource allocation for mobile edge computing with multiple access points," *IET communications*, vol. 13, no. 17, pp. 2668–2677, 2019.

[120] H. Wang and J. Xie, "You can enjoy augmented reality while running around: An edge-based mobile AR system," in *Proceedings of ACM/IEEE Symposium on Edge Computing (SEC)*, (San Jose, CA), pp. 381–385, 2021.

[121] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, January 2019.

[122] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (Paris, France), pp. 1423–1431, 2019.

[123] C. Dong, S. Hu, X. Chen, and W. Wen, "Joint optimization with DNN partitioning and resource allocation in mobile edge computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 3973–3986, December 2021.

[124] P. Dong, Y. Xia, L. Zhuo, and D. Feng, "Real-time moving object segmentation and tracking for H.264/AVC surveillance videos," in *Proceedings of IEEE International Conference on Image Processing*, pp. 2309–2312, 2011.

[125] M. Makar, V. Chandrasekhar, S. S. Tsai, D. Chen, and B. Girod, "Interframe coding of feature descriptors for mobile augmented reality," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3352–3367, 2014.

[126] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 1–16, 2019.

[127] S. Huang and J. Xie, "DAVE: Dynamic adaptive video encoding for real-time video streaming applications," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2021.

[128] Z. Gao, W. Hao, Z. Han, and S. Yang, "Q-learning-based task offloading and resources optimization for a collaborative computing system," *IEEE Access*, vol. 8, pp. 149011–149024, August 2020.

[129] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, February 2018.

[130] S.-Y. Han and H.-W. Lee, "Deep reinforcement learning based edge computing for video processing," *ICT Express*, May 2022.

[131] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, February 2019.

[132] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10843–10856, July 2021.

[133] J. Wang, "An intuitive tutorial to Gaussian processes regression," *arXiv preprint arXiv:2009.10862*, 2020.

[134] S. Huh, S. Muralidharan, H. Ko, and B. Yoo, "XR collaboration architecture based on decentralized web," in *Proceedings of ACM International Conference on 3D Web Technology (Web3D)*, (Los Angeles, CA), pp. 1–9, 2019.

[135] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Vancouver, British Columbia, Canada), pp. 7464–7475, 2023.

[136] D. Kim, S. Song, J. Lee, W. Cui, W. Choi, and S. An, "Dynamic location management scheme for location registers in wireless ATM networks," *Electronics Letters*, vol. 36, pp. 1650–1651, September 2000.

[137] J. Xie, I. Howitt, and I. Shibeika, "IEEE 802.11-based mobile IP fast handoff latency analysis," in *Proceedings of IEEE International Conference on Communications (ICC)*, (Glasgow, UK), pp. 6055–6060, 2007.

[138] V. Solouk, B. M. Ali, and K. D. Wong, "Vertical fast handoff in integrated WLAN and UMTS networks," in *Proceedings of International Conference on Wireless and Mobile Communications (ICWMC)*, (Luxembourg), pp. 59–64, 2011.

[139] H. Arbabi and M. C. Weigle, "Highway mobility and vehicular ad-hoc networks in ns-3," in *Proceedings of IEEE Winter Simulation Conference*, 2010.

[140] "NVIDIA Jetson AGX Xavier developer kit." https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.

[141] "LinkSys WRT1900ACS dual-band Wi-Fi router." https://www.linksys.com/support-product?sku=WRT1900ACS.

[142] "Monsoon power monitor." https://www.msoon.com/specifications.

[143] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[144] "COCO: Common objects in context." https://cocodataset.org/home.

[145] K. Suri, "Off-policy evolutionary reinforcement learning with maximum mutations," in *Proceedings of ACM/SIGAI International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (Auckland, New Zealand), pp. 1237–1245, 2022.

[146] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of International Conference on Machine Learning (ICML)*, (Stockholm, Sweden), pp. 1861–1870, 2018.

[147] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[148] D. M. Casas-Velasco, O. M. C. Rendon, and N. L. da Fonseca, "Intelligent routing based on reinforcement learning for software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 870–881, March 2020.

[149] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[150] "Tensorflow Lite." https://www.tensorflow.org/lite.

[151] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV), pp. 779–788, 2016.

[152] "Cisco OpenH264." http://www.openh264.org/index.html.

[153] S. Barcellona and L. Piegari, "Effect of current on cycle aging of lithium ion batteries," *Journal of Energy Storage*, vol. 29, p. 101310, June 2020.

[154] J. Lee, Y. Chon, and H. Cha, "Evaluating battery aging on mobile devices," in *Proceedings of ACM Annual Design Automation Conference (DAC)*, (New York City, NY), pp. 1–6, 2015.