AWS-Based Edge Computing Optimization for Industrial IoT Video Streams

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Software Engineering
March 2025
Tommy Krook

UNIVERSITY OF TURKU
Department of Computing

Tommy Krook: AWS-Based Edge Computing Optimization for Industrial IoT Video Streams

Master of Science (Tech) Thesis, 73 p., 1 app. p.
Software Engineering
March 2025

Industrial Internet of Things (IIoT) systems increasingly rely on video streams for real-time monitoring, quality control, and predictive maintenance. However, processing these video streams efficiently presents significant challenges, particularly regarding latency, bandwidth consumption, and computational costs. This thesis explores the trade-offs between edge and cloud computing for IIoT video stream processing, with a focus on optimizing performance while minimizing operational costs.

The research is based on a series of controlled experiments that compare edge-only, cloud-only, and hybrid computing approaches. The experiments evaluate key performance metrics such as latency, processing speed, and bandwidth utilization. Results indicate that while cloud computing offers high processing power, it introduces significant network delays and higher data transmission costs. Conversely, edge computing reduces latency and decreases bandwidth usage but is limited by the processing constraints of local devices. A hybrid approach, where preprocessing occurs at the edge before sending refined data to the cloud for further analysis, proves to be the most efficient solution.

This study demonstrates that distributing computational tasks strategically between edge and cloud environments can enhance IIoT video analytics. By preprocessing video streams at the edge, the amount of data transmitted to the cloud is significantly reduced, leading to improved efficiency and cost savings. The findings contribute to ongoing research on IIoT optimization, providing insights into how intelligent workload distribution can improve industrial video stream processing in smart manufacturing and other IIoT applications.

Keywords: AWS, Edge Computing, IIoT, Object Detection, Video Stream

Contents

List of Figures

List of Tables

List of acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

API Application Programming Interface

AWS Amazon Web Services

CNN Convolutional Neural Network

COCO Common Object in Context

CPU Central Processing Unit

DDoS Distributed Denial-of-Service

DL Deep Learning

DNN Deep Neural Network

EI Edge Intelligence

EoL End of Life

GCP Google Cloud Platform

GPU Graphics Processing Unit

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a Service

IIoT Industrial Internet of Things

IoT Internet of Things

ML Machine Learning

PaaS Platform as a Service

QoE Quality of Experience

RAN Radio Access Network

RTSP Real-Time Streaming Protocol

SaaS Software as a Service

SDK Software Development Kit

SGD Stochastic Gradient Descent

SLA Service Level Agreement

TCP Transmission Control Protocol

TPU Tensor Processing Unit

UDP User Datagram Protocol

YOLO You Only Look Once

1 Introduction

Video streaming brings many benefits to the ecosystem of Industrial Internet of Things (IIoT). Video streams from cameras and other visual sensors are delivering essential insights into industrial operations such as real-time monitoring, predictive maintenance, quality control, process optimization and improved security. Video data adds a valuable visual layer to traditional sensor information, offering a deeper and more complete understanding of the industrial environment. [1]

Edge streaming is a network architecture focused on processing and streaming video data directly at the network's edge, close to the data source, often where cameras or sensors capture the video. This strategy reduces the need for data to travel back and forth to distant data centers or cloud servers. [2], [3] In contrast, cloud streaming involves transmitting video data from edge devices, such as cameras or sensors, to a centralized cloud server, where it is processed and stored before being streamed to end-users or other devices. [1] This thesis focuses on edge streaming and cloud streaming, and primarily to the combination of those two. Often in the literature this case is referred to as a hybrid approach.

The global edge computing market is expected to grow to 350 billion U.S. dollars by 2027 [4]. IoT has the potential to create an annual economic impact of between $3.9 trillion and $11.1 trillion by 2025. [5].

1.1 Research Objectives and Scope

The objective of this thesis is to examine the impact of combining edge and cloud computing on video stream processing efficiency, latency, and bandwidth usage within IIoT environments, particularly for real-time video analytics. The study will focus exclusively on Amazon Web Services (AWS) as the cloud platform due to my familiarity with it, as including additional platforms, such as Microsoft Azure or Google Cloud Platform (GCP), would significantly broaden the scope and length of the thesis.

The experiments will be conducted using a Raspberry Pi 5 with 8GB RAM as the edge device. Although there are various other edge devices available, I have chosen this one because it is accessible to me, and expanding to multiple equally powerful devices would likely not affect the outcome of this study.

This research will leverage existing machine learning models to perform video analytics rather than training custom models. This approach aligns with the objective of examining the computational and resource distribution aspects of edge and cloud computing without delving into model development. Also this thesis will not study the accuracy of the models. Additionally, I will utilize existing software libraries and frameworks wherever possible to streamline the implementation and focus on the core objectives of this thesis. It is important to note also that this study will not explore the impact of network problems.

My research specifically targets the IIoT sector rather than general IoT, as IoT is a broad field encompassing a wide range of applications. My experiments and simulations will concentrate on scenarios relevant to "smart factories", a subset of IIoT, while acknowledging that IIoT itself spans even broader applications. Covering all these applications would extend the thesis beyond its intended focus. However, I will include examples from general IoT and IIoT in the literature review for broader context.

Finally, this thesis will focus on edge and cloud computing, excluding other paradigms such as fog computing, to maintain a clear and manageable scope.

1.2 Research Questions

From that research scope, I have come up with these research questions:

RQ1: What are the trade-offs between edge and cloud computing for processing industrial video streams in terms of cost and performance?

RQ2: What type of video stream processing tasks should be distributed between the edge and the cloud for optimal performance in industrial IoT systems?

RQ3: How does video preprocessing at the edge impact communication overhead in industrial IoT systems?

1.3 Thesis Structure

This thesis is structured into six chapters. The first chapter provides an introduction, offering a brief overview of the topic. It outlines the research objectives, scope, and key research questions. Additionally, it defines the structure of the thesis and details the research methodologies employed.

The second chapter explores fundamental concepts relevant to this study, including the Industrial Internet of Things (IIoT), edge computing, and cloud computing. It discusses the benefits and challenges associated with these technologies and highlights key Amazon Web Services (AWS) offerings relevant to the research.

The third chapter introduces machine learning, with a deeper focus on deep learning as a subset of the field. It examines various approaches to implementing machine learning at the edge and in the cloud. For example, it discusses open-

source computer vision solutions for edge computing and AWS SageMaker AI for cloud-based applications.

The fourth chapter reviews existing studies related to this research. It further elaborates on the research questions and defines the edge hardware used in this study, specifically the Raspberry Pi 5 with 8 GB of RAM. The chapter also describes the experimental setup, including experiment definitions, and addresses challenges encountered before conducting the experiments.

The fifth chapter presents an analysis of the experimental results and evaluates the cost structures of different AWS services. The sixth chapter discusses shortly, how the research questions have been answered. Finally, the seventh chapter concludes the thesis by summarizing key findings and presenting suggestions for further research.

1.4 Research Methodologies

In this thesis, I employ a structured approach that aligns with the decision-making framework presented in [6]. The framework includes three phases: strategy, tactical and operational phase.

Research Strategy Phase

The research strategy phase consists of four parts: outcome, logic, purpose and approach.

Research Outcome: The research is classified as applied research. The primary goal is to solve a specific, real-world problem by leveraging existing theories and technologies. This decision aligns with the necessity of providing actionable results for industry practitioners. This includes for example the usage of existing object detection models and streaming libraries.

Research Logic: A deductive research approach is utilized. This choice stems

from the need to test hypotheses derived from established theories, ensuring a robust and scientific validation of results.

Research Purpose: The purpose is both descriptive and explanatory. Descriptive aspects aim to characterize the current state of combining edge and cloud computing, while explanatory efforts focus on understanding causal relationships and mechanisms underlying observed phenomena. This is reflected in the literature review and in the evaluation of the results.

Research Approach: The thesis follows a positivist research approach. This decision reflects the objective nature of the study, which relies on quantifiable data and reproducible experiments to establish generalizable findings. I will try to describe my experiments in a way that they could be reproducible.

Research Tactical Phase

The research tactical phase consists of two parts: process and methodology.

Research Process: The study adopts a quantitative research process. This method ensures the results are measurable, statistically analyzable, and comparable.

Research Methodology: A case study research methodology is applied, focusing on a specific scenario or system relevant to the thesis objectives. This approach enables an in-depth exploration of the research problem within a real-world context. Literature review is used to collect information about the current state of this topic.

Research Operational Phase

The research operational phases describes data collection methods and data analysis methods.

Literature Review: A comprehensive review of existing literature establishes the theoretical foundation and identifies gaps that the research aims to address.

Experiments: Controlled experiments are conducted to evaluate specific as-

pects of the system or process being studied. These experiments help in isolating variables and understanding their impacts.

Simulations: Simulations are employed to replicate scenarios that are impractical or costly to observe in real-world settings, enabling the study of system behavior under varying conditions.

Statistics: Statistical techniques are applied to analyze experimental and simulated data, ensuring rigorous validation of results.

The decision to use applied research stems from the practical implications of the research topic, which requires solutions that can be directly implemented in Industrial Internet of Things (IIoT) environment. A deductive approach ensures that findings are grounded in established theories, providing a solid foundation for conclusions. The dual focus on descriptive and explanatory purposes allows for a holistic understanding of both the current state and the dynamics of the research topic.

The quantitative process was chosen due to its ability to provide objective and reproducible results, which are crucial for the validity of the research. A case study methodology enables a deep dive into specific scenarios, offering detailed insights that are often unattainable through broader surveys or observational studies.

All of these combined, will try to answer the research questions but there are some parts that will have the most significant effect. The research operational phase will play critical role on finding the answers to the research questions. Literature review will describe the current state of the topic and identify research gaps, it will also partly answer all of the research questions. Experiments and simulations will complement the results of the literature review.

2 Industrial Internet of Things

The Internet of Things (IoT) is a flexible global network infrastructure that uses standardized, compatible communication protocols to allow devices to automatically connect and interact with one another. This setup enables seamless information exchange and supports collaborative decision-making among connected devices. [7] In IoT, a variety of objects with unique addresses or identities are linked through diverse transmission networks, enabling dynamic information exchange among them [8]. Applying IoT in the industrial sector has led to the development of the Industrial Internet of Things (IIoT), a new research field born from IoT's success in creating "smart" factories [8], [9].

The most common application scenarios for IoT include smart homes, health monitoring, indoor localization, and more [10], [11]. It aims to improve the quality of life [8]. The IIoT is focused on enhancing industrial production processes, with the goal of increasing efficiency in manufacturing and operations. Common application scenarios for the IIoT include remote maintenance, smart logistics and intelligent factories [8], [9]. Other areas where IIoT is used are for example smart grids [8], environment monitoring, agriculture, health care [12] and smart cities [13].

IIoT utilizes technologies such as Artificial Intelligence (AI), Machine Learning (ML), big data analytics, edge computing, and cloud computing to analyze and make sense of the extensive data produced by industrial devices. [14]

2.1 Architecture and Standardization of IIoT

A collective IIoT stack (Figure 2.1), which outlines the technology layers in an IIoT system, typically includes layers such as devices/sensors, connectivity, data processing, and applications. Starting with the device layer, which includes the physical devices and sensors that gather data from the industrial environment. The connectivity layer follows, encompassing communication technologies that facilitate data transfer between devices and data centers or cloud. Next is the data processing layer, which covers technologies for data processing, storage and analysis, often through cloud and edge computing solutions. Finally, the application layer involves applications and services that leverage processed data to provide insights and support decision-making. [15]

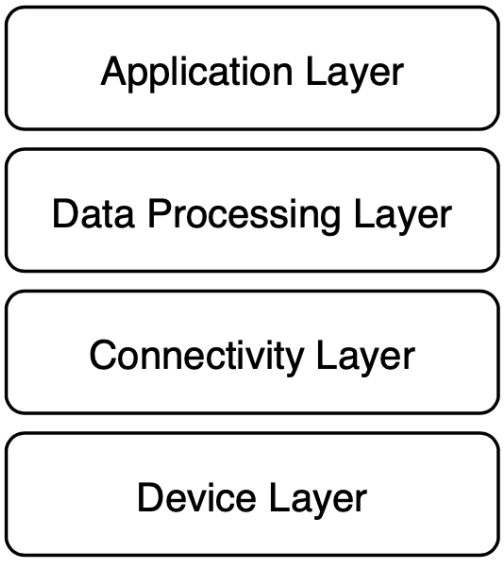

Figure 2.1: A collective IIoT stack.

IIoT faces critical challenges with interoperability due to the wide range of devices, communication protocols, and data formats it employs. While various standards and protocols have been developed to bridge these gaps, a truly unified ap-

proach is still missing. This diversity in IIoT systems leads to significant integration difficulties, often resulting in vendor lock-in, where customers become dependent on a specific vendor's ecosystem and unable to easily switch between platforms. [16]

2.2 Edge Computing

Edge computing differs from traditional cloud computing as it introduces a new paradigm that processes data at the network's edge. The primary concept is to bring computation closer to the data source, minimizing the distance between data generation and processing. [3] Definitions of edge computing varies among researchers and technology vendors. Professor Satyanarayanan defines it as

> "*Edge computing* is a new computing model that deploys computing and storage resources (such as cloudlets, micro data centers, or fog nodes, etc.) at the edge of the network closer to mobile devices or sensors" [3].

AWS defines it as

> "*Edge computing* is the process of bringing information storage and computing abilities closer to the devices that produce that information and the users who consume it." [17].

In the context of edge computing, fog computing is also often mentioned. Let us define it here, so we can understand their difference. The fundamental distinction between edge and fog computing lies in the specific location where computation occurs [18]. Fog computing is a virtualized system that offers computing, storage, and networking services situated between end devices and cloud data centers, often positioned close to the network's edge [19]. Fog computing is a model that shifts tasks from central cloud data centers to lightweight servers positioned closer to devices. This is achieved by introducing a fog layer, which comprises servers situated between the cloud and edge devices [20].

2.2.1 Why is Edge Computing Needed

There are limitations on cloud services, while cloud computing is powerful, network bandwidth has not kept pace with data generation. Transmitting all data to the cloud can create bottlenecks, making it inefficient for applications that need real-time processing, like autonomous vehicles.

The amount of IoT devices is growing, with billions of IoT devices generating large amounts of data, cloud resources alone can not handle the load. Processing data at the edge reduces the need for constant cloud transmission, lowers network strain, conserves energy, and enhances data privacy.

Devices are shifting from consumers to producers, as more devices both produce and consume data, like smartphones uploading videos, handling data at the edge helps reduce bandwidth usage, allowing data to be adjusted or filtered locally before it reaches the cloud, which improves efficiency and response times. [2]

2.2.2 Benefits

In edge computing, the goal is to place computing resources close to the data sources. This approach offers several advantages over the traditional cloud-based computing model. [2] Researchers developed a proof-of-concept platform for running a facial recognition application. By shifting computation from the cloud to the edge, they were able to reduce the response time from 900 milliseconds to 169 milliseconds. [21] Researchers utilized cloudlets to offload computing tasks for wearable cognitive assistance, resulting in a response time improvement of between 80 and 200 milliseconds [22].

Reducing operational costs is always beneficial. Using edge devices instead of cloud services can lower costs, thanks to reduced operational and management expenses. Edge computing accelerates response times and optimizes time to action, all while saving network resources. It minimizes latency and alleviates network con-

gestion. Edge devices provide greater scalability and improved energy efficiency for IoT systems. [23] For instance, in industrial IoT, machinery equipped with sensors can analyze data on-site to detect anomalies or predict maintenance needs without relying on cloud processing. By handling this data locally, energy is saved, allowing for more compact, efficient devices that seamlessly integrate into industrial environments like factory floors or warehouses. With edge devices, data analysis is nearly real-time, as it occurs locally rather than in a distant data center or cloud [23].

Context-aware computing is also becoming increasingly significant in IoT and edge computing applications, as effective data modeling and reasoning often depend on understanding the context in which data is collected. Due to their proximity to data sources, edge servers are well-positioned to gather richer contextual information, enhancing their ability to support data processing. [24] For example, in Amazon Go stores, video cameras do not just capture the products customers pick up, they also assess customer interests by tracking where customers pause, how long they spend in certain areas, and their browsing behaviors. [25]

Based on [26]–[28], edge computing offers several advantages, particularly for video streaming services. It enables low latency by placing computation and storage resources closer to data source, meeting strict requirements for quick response times. Additionally, it reduces bandwidth costs by allowing local data downloads, which eliminates the need for constant data exchange between cloud servers and edge device. Moreover, for streaming services outside of IIoT, edge computing enhances the quality of experience (QoE) by using radio access network (RAN) data, such as network load and user location, to support smarter video adaptation, which boosts user satisfaction. Furthermore, it enables targeted content delivery by caching popular videos on local edge servers, improving QoE for different regions and reducing redundant data storage.

2.2.3 Challenges

There are also challenges in edge computing. IoT devices have limited processing capabilities, while complex enterprise calculations demand substantial processing power. There might be a need for huge storage capacity but usually IoT devices do not have that. Availability might be an issue also. Edge devices may sometimes be unable to connect to the cloud due to factors such as network congestion, limited coverage, or system failures. [23]

Challenges in Security

Based on [29], the main types of attacks that can directly target edge computing applications fall into six categories: side-channel attacks, Distributed Denial-of-Service (DDoS) attacks, malware injection, authentication and authorization attacks, man-in-the-middle attacks, and bad-data injection attacks. The percentage decomposition can be seen below (Figure 2.2)

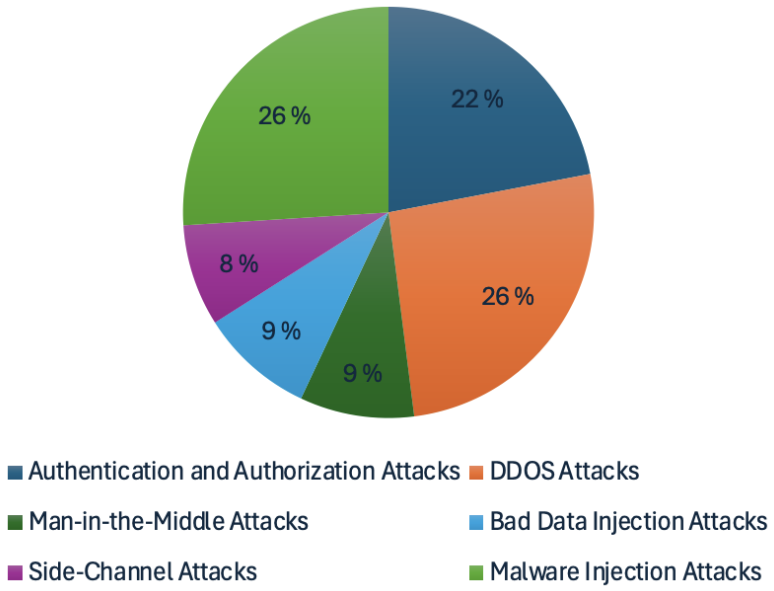

Figure 2.2: The distribution of attack types targeting edge computing infrastructures based on real-world occurrences. [29]

The expansion of edge computing introduces more security risks by increasing the attack surface in several ways. First, edge servers have limited processing power compared to cloud servers, making them more vulnerable to attacks that may not affect cloud infrastructure. Additionally, edge devices tend to have weaker defenses than general-purpose computers, leaving them open to threats that standard systems could easily handle.

Second, many IoT devices lack user interfaces, which limits users' ability to monitor their status. This lack of awareness means that if an edge device is compromised, users are often unable to detect it.

Third, edge devices operate with a diverse range of operating systems and protocols, unlike standard computers that typically use common systems and protocols. This variation complicates the development of universal security solutions for edge environments.

Finally, existing access control models, which were designed for traditional computing systems, are often too simplistic for the complex requirements of edge computing. Edge systems require more refined access controls to manage intricate permissions around who can access certain sensors and under what conditions, yet most current models are too basic for these needs. [29]

2.3 Cloud Computing

2.3.1 What is Cloud Computing

Cloud computing allows users to access various IT services, like applications, storage, and networking, on a pay-as-you-go basis over the internet. These services are hosted remotely by a provider, so users do not need to install or maintain any software or infrastructure locally. This setup helps users quickly scale their usage up or down as needed without upfront investments in hardware or software.

The term "cloud computing" includes three main service models (Figure 2.3):

1. **Software as a Service (SaaS)**: This allows users to operate applications without managing the underlying system, hardware, or network on which the applications run.

2. **Platform as a Service (PaaS)**: This provides a hosting environment for users to manage and develop applications, without dealing with the underlying system, hardware, or infrastructure.

3. **Infrastructure as a Service (IaaS)**: This offers virtual computing resources, like processing power, storage, and network capabilities, without giving users control over the underlying cloud infrastructure. [30]–[33]

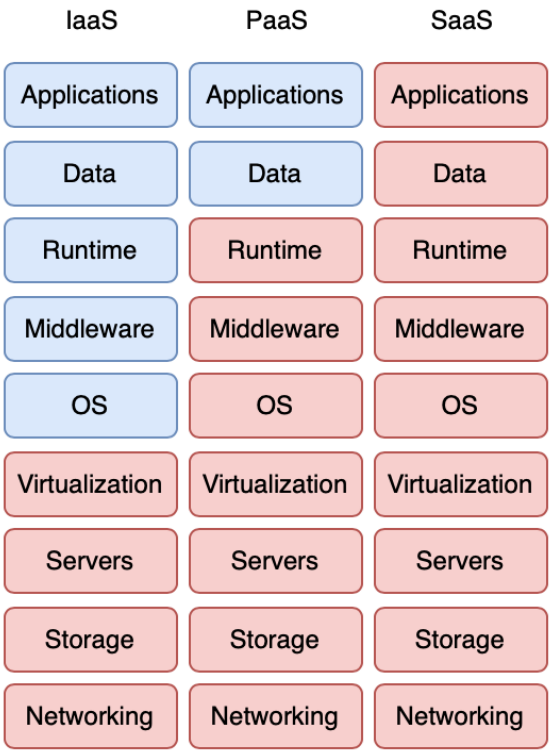

Figure 2.3: Three main service models for cloud computing.

There are four main types for cloud deployments [30], [31]:

1. **Private cloud**: This type of cloud is used exclusively by one organization.

2. **Public cloud**: A company provides this cloud for general public use or to a large group, offering cloud services on a broad scale.

3. **Community cloud**: This cloud is shared by multiple organizations with similar interests or needs.

4. **Hybrid cloud**: This combines two or more clouds (private, public, or community), keeping them separate but allowing data and applications to move between them.

2.3.2 Benefits

Cloud computing brings many benefits, e.g. cost savings, scalability, QoE, reliability, effiency and security [30], [34]. Costs can be reduced by enabling companies to bypass expensive asset purchases and decrease maintenance costs. Since cloud services are available on-demand, organizations pay only for the services they use when they need them.

Resources and services in cloud computing can be swiftly and flexibly adjusted, allowing for quick scaling up or down as needed. For consumers, these resources often seem limitless and are available for purchase in any quantity at any time. Cloud computing allows resource usage to be tracked, managed, and reported, offering clear visibility for both the provider and the consumer regarding the services used.

Data and user activities are replicated almost in real-time across multiple globally distributed data centers. If one data center becomes unavailable, the system is designed to immediately switch to a backup center, ensuring a seamless experience without any noticeable service disruption for users. For example, in the AWS's Service Level Agreement (SLA) they promise an uptime of at least 99,99% per month [35].

When services are shifted to the cloud, the need for in-house support and service functions is significantly reduced. Cloud computing enhances security through built-in measures like data redundancy, encryption, and continuous monitoring.

2.3.3 Challenges

According to [36], compliance and control over data present significant hurdles for cloud adoption. Organizations worry about meeting legal and regulatory requirements when using cloud platforms, especially regarding data stored across multiple jurisdictions. Additionally, concerns about losing control over data in public cloud environments make organizations cautious about cloud adoption, favoring private clouds where possible, despite higher costs.

The paper also highlights the organizational difficulty of transitioning to cloud computing. These challenges include a lack of skilled human resources capable of managing cloud integration and resistance to adopting new technologies. Participants also cite funding constraints and the need for more extensive organizational support as additional barriers.

According to [37], one challenge of cloud is governance control. It is important to ensure effective management and oversight of organizational assets. It is essential to have a dedicated team in place to monitor and verify that these assets are utilized in compliance with established policies and procedures. Proper maintenance and strategic use of these resources are crucial to supporting the organization in achieving its objectives.

2.3.4 Amazon Web Services

AWS is the most extensive and widely used cloud platform worldwide, providing over 200 complete services through data centers around the globe. AWS provides a much broader selection of services and features than any other cloud provider. Its offer-

ings range from foundational technologies like computing, storage, and databases to cutting-edge solutions in artificial intelligence, machine learning, data lakes, analytics, and the IoT. This extensive range makes it easier, faster and more cost-effective to migrate existing applications to the cloud and create almost anything you envision.

In addition, AWS offers the most in-depth functionality across its services. For instance, it provides a diverse array of specialized databases designed for various applications, allowing you to select the best-suited tool to optimize both cost and performance. [38] Next I will introduce few services that I will be using in this thesis.

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) offers flexible, on-demand computing power in the AWS cloud, helping to cut hardware costs and accelerate application deployment and development. With Amazon EC2, you can launch any number of virtual servers, configure security and networking, and manage storage. You can scale up to meet high-demand tasks like periodic processing or traffic spikes and scale down as demand decreases. [39]

Amazon Simple Storage Service

Amazon Simple Storage Service (Amazon S3) is a versatile object storage solution designed to deliver exceptional reliability, scalability, security, and performance. Suitable for businesses of all sizes and industries, Amazon S3 allows users to store and safeguard unlimited data for a wide array of applications, including websites, data lakes, mobile apps, archives, backups, enterprise solutions, IoT data, and big data analytics. S3 includes robust management tools that help you optimize and organize your data while configuring access controls to align with your unique busi-

ness, compliance, and organizational needs. This flexibility makes it a powerful choice for a broad range of storage use cases. [40]

AWS IoT Core

AWS IoT offers cloud services to link your IoT devices with other devices and AWS cloud services. It also provides device software to help integrate your IoT devices into AWS IoT-based solutions. Once your devices are connected to AWS IoT, they can access the full range of cloud services available through AWS. [41]

AWS IoT Greengrass

AWS IoT Greengrass is an open-source edge runtime and cloud service for the IoT that allows you to build, deploy, and manage IoT applications directly on your devices. With AWS IoT Greengrass, you can create software that enables devices to process data locally, make predictions with machine learning models, and filter or aggregate data. This service lets devices gather and analyze data close to its source, respond independently to local events, and communicate securely with other devices on the same network. Greengrass-enabled devices can also connect securely to AWS IoT Core, sending IoT data to the AWS Cloud. Additionally, you can develop edge applications using pre-configured software components that link your edge devices to AWS or third-party services. [42]

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams is a fully managed AWS service for streaming live video from devices to the cloud, enabling real-time video processing or batch video analytics. Beyond simple storage, Kinesis Video Streams lets you monitor live streams in the AWS management console or through custom applications using its Application Programming Interface (API). The service can handle vast amounts

of live video from various sources like smartphones, security cameras, and drones, as well as other time-serialized data like audio and thermal imagery.

Kinesis Video Streams securely stores and encrypts media data with time indexing, allowing you to process video data in real time or access historical data as needed. You can run applications on Amazon EC2 to analyze the video streams using machine learning algorithms or third-party integrations. [43]

Simple Notification Service

Amazon Simple Notification Service (Amazon SNS) is a managed service that facilitates message delivery between publishers and subscribers, often referred to as producers and consumers. It allows publishers to send messages asynchronously to a "topic", which acts as a virtual channel and access point for communication.

Subscribers can connect to an Amazon SNS topic and receive messages through various supported endpoint types. These include services like Amazon Data Firehose, Amazon SQS, AWS Lambda, as well as Hypertext Transfer Protocol (HTTP), email, mobile push notifications, and SMS messages. [44]

Simple Queue Service

Amazon Simple Queue Service (Amazon SQS) is a reliable, secure, and highly available cloud-based queueing service designed to help you integrate and decouple different components of distributed software systems. It includes features like dead-letter queues and cost allocation tags to enhance functionality and management. Amazon SQS provides a versatile web services API that can be accessed using any programming language supported by the AWS Software Development Kit (SDK). [45]

Lambda

AWS Lambda allows you to execute code without the need to set up or manage servers. It automatically handles the underlying compute infrastructure, including server maintenance, operating system updates, capacity management, scaling, and logging. All you need to do is provide your code using one of the supported language runtimes.

Your code is structured as Lambda functions, which are executed only when triggered. The service scales automatically to meet demand and charges you only for the compute time your functions use. [46]

3 Machine Learning

Because this thesis utilizes object detection tools, it is important to provide a brief overview of machine learning as a foundation. Understanding the basic principles of machine learning (ML) helps contextualize the methods used for video stream processing, particularly in edge and cloud computing environments. This background ensures a clearer perspective on how ML-driven object detection enhances efficiency and decision-making in IIoT applications.

Defining ML clearly can be a bit challenging. The following is based on [47]. ML is a form of artificial intelligence (AI) that enables machines to learn from data without the need for explicit programming. This is achieved by adjusting model parameters through calculations so that the model's behavior adapts to the given data or experiences. The learning algorithm continually refines these parameter values, allowing the ML model to learn over time and make informed predictions or decisions based on data analysis.

ML has diverse applications across various industries, including healthcare, finance, marketing, and transportation. Today, ML models are actively used in areas like fraud detection, image recognition, natural language processing, recommendation engines, autonomous vehicles, and personalized medicine.

Overall, ML is key to enabling computers to learn from data and experience, improving their performance on specific tasks without manual programming. It has the potential to transform many industries by automating complex tasks and

making intelligent decisions through the analysis of vast amounts of information.

ML builds upon core computer science concepts, relying significantly on statistics, probability, and optimization techniques. There are three primary categories of ML models:

1. **Supervised Learning**: Supervised learning is used for predicting outcomes or classifying data based on labeled training datasets. As data is fed into the model, it undergoes a cross-validation process to adjust the model's weights until it reaches a suitable fit. This type of learning supports applications such as face recognition, object detection, and quality control.

2. **Unsupervised Learning**: In contrast to supervised learning, unsupervised learning deals with unlabeled datasets. Its goal is to help ML models discover hidden patterns or structures without the need for human oversight. Businesses often use unsupervised learning for cross-selling strategies, customer segmentation or exploratory data analysis.

3. **Reinforcement Learning**: Reinforcement learning, similar to supervised learning, relies on trial-and-error techniques. Instead of using labeled training datasets, reinforcement learning enables models to learn through iterative processes, adapting their behavior based on successful outcomes. This helps train models to provide the best possible recommendations or actions over time.

3.1 Deep Learning

Deep learning (DL) is a branch of ML that focuses on training artificial neural networks. These networks are composed of multiple layers and are inspired by the structure and functioning of the human brain. Similar to our brains, they are made up of interconnected nodes (or neurons) that transmit signals.

DL algorithms are particularly powerful in areas like image and speech recognition, natural language processing, and more. They automatically extract meaningful features from raw data by passing it through multiple layers of abstraction. This approach makes DL effective in working with large-scale datasets and complex, high-dimensional inputs. However, it requires substantial computational power and extensive training to achieve these capabilities. [47]

3.2 Deep Learning Methods Generally

Deep Neural Networks

Deep neural networks (DNNs), often referred to as deep fully connected neural networks, stand apart from traditional artificial neural networks (ANNs) due to their deeper and more intricate layer structure, enabling them to handle more complex learning tasks [48]. A typical DNN is composed of an input layer, multiple hidden layers, and an output layer, where the output from each layer is passed to the next using activation functions. The final layer produces the model's prediction. Training these networks often involves optimization methods like Stochastic Gradient Descent (SGD) and backpropagation, which refine the network's weights for improved performance. DNNs are widely applied in areas such as feature extraction, classification, and function approximation, showcasing their powerful adaptability to diverse applications. [25]

Convolutional Neural Networks

Convolutional neural networks (CNNs) are specifically designed to handle data structured as multiple arrays. For instance, a color image is represented by three 2D arrays, each corresponding to pixel intensity values for the three color channels. [49] CNN processes two-dimensional data by identifying patterns and important features.

It does this using special filters that scan through the data, capturing relationships between nearby elements. After this, a pooling step simplifies the information by reducing its size while retaining essential details. This approach helps CNNs recognize patterns more efficiently than traditional deep learning models, as it reduces the number of required parameters and minimizes the risk of overfitting. Because of these advantages, CNNs are widely used in applications like object detection and health monitoring. [25]

There are other methods as well. Some of these are for example restricted Boltzmann machine, autoencoder, recurrent neural networks and deep reinforcement learning.

3.3 Deep Learning Methods for Video Streams in Edge Device

3.3.1 TensorFlow

TensorFlow is a system developed to handle large-scale machine learning tasks. It represents computations, shared states, and operations through dataflow graphs, enabling efficient distribution across computational devices like CPUs, GPUs, and Tensor Processing Units (TPUs). Its flexibility enables the development, training, and deployment of machine learning models across a wide range of environments, from large distributed clusters to mobile devices. TensorFlow has been instrumental in advancing research and production-level applications in machine learning, making it a cornerstone for modern AI development. [50]

LiteRT, formerly known as TensorFlow Lite, is Google's high-performance runtime designed for executing machine learning models directly on devices such as smartphones, embedded systems, and microcontrollers. It enables efficient on-device

AI by addressing key constraints like latency, privacy, connectivity, size, and power consumption. LiteRT supports models from various machine learning frameworks, including TensorFlow, PyTorch, and JAX, allowing developers to convert and run these models in the .tflite format using AI Edge conversion and optimization tools. [51]

3.3.2 Keras and MobileNet

Keras is an open-source deep learning API written in Python, designed to simplify the creation and training of neural networks. It operates as a high-level interface, running on top of frameworks like TensorFlow, JAX, or PyTorch, thereby providing flexibility and compatibility across different machine learning ecosystems. [52]

Keras applications offer a collection of pre-built deep learning models that come with pre-trained weights. These models can be utilized for tasks such as making predictions, extracting features, or refining them further for specific use cases. [53] One of these applications is MobileNet. MobileNet is a class of lightweight and efficient convolutional neural networks designed for mobile and embedded vision applications. These models are built using a streamlined architecture that leverages depthwise separable convolutions. This approach reduces computation and model size significantly compared to standard convolutional networks, making them ideal for resource-constrained environments like mobile devices and edge computing. [54]

3.3.3 Open Source Computer Vision

OpenCV, short for Open Source Computer Vision Library, is a freely available software library designed for machine learning and computer vision. Its primary goal is to offer a shared platform for developing computer vision applications and to promote the integration of machine perception into commercial products. Licensed under Apache 2.0, OpenCV allows businesses to easily access, use, and customize

its code.

The library features over 2,500 optimized algorithms that cover a wide range of functionalities, from classic to cutting-edge computer vision and machine learning techniques. These capabilities include face detection and recognition, object identification, human action classification in videos, camera motion tracking, moving object tracking, generating 3D point clouds from stereo cameras, 3D object modeling, stitching images to create high-resolution panoramic views, searching for similar images in a database, correcting red-eye in photos, monitoring eye movements, recognizing scenes, and enabling augmented reality by placing markers on visual elements. OpenCV boasts a vibrant user community of more than 47,000 members and has been downloaded over 18 million times. It is widely utilized by businesses, research institutions, and government organizations. [55]

3.3.4 You Only Look Once: Unified, Real-Time Object Detection

You Only Look Once (YOLO) is a real-time object detection system that revolutionizes the way object detection is performed. Unlike traditional methods that rely on complex multi-step pipelines (e.g., region proposals and subsequent classification), YOLO simplifies the process by treating object detection as a single regression problem. This unified approach predicts bounding boxes and class probabilities directly from the full image in a single step using a convolutional neural network (CNN).

YOLO's architecture is designed for speed and efficiency, making it capable of processing images at remarkable real-time rates. It divides an input image into a grid, where each grid cell is responsible for predicting bounding boxes and their associated confidence scores, as well as the class probabilities for detected objects. This grid-based approach enables YOLO to capture both spatial and contextual information, reducing the likelihood of false detections and enhancing its robustness

in a variety of scenarios. [56]

3.3.5 Common Object in Context

Common Objects in Context (COCO) is a comprehensive dataset designed to advance computer vision research by focusing on the detection, segmentation, and contextual understanding of objects in everyday scenes. Unlike earlier datasets that often showcased objects in isolated or canonical perspectives, COCO emphasizes non-iconic images, where objects appear in complex, natural environments with occlusion, clutter, and diverse viewpoints.

COCO has become a gold standard for evaluating and benchmarking object detection and segmentation algorithms. It provides a strong foundation for pushing the boundaries of scene understanding, contextual reasoning, and the development of more generalizable computer vision models. [57]

Now let us think of a little analogy and that way understand, how for example OpenCV, YOLO and COCO come together. Think of OpenCV, YOLO, and the COCO library as parts of a movie production team. OpenCV is the film studio, providing all the essential equipment and setup needed to shoot and edit a movie. YOLO is the director, actively analyzing the scenes and pointing out who or what is present in real time, like identifying actors or props. The COCO library is the script or storyboard that guides the director, helping YOLO recognize and name the elements in the scene. Without the studio, the director could not work, and without the script, the director would not know what to look for. Together, they create a seamless process of capturing and interpreting the visual world.

While I am discussing these methods in the context of edge computing in this subchapter, it is important to note that they can also be utilized in cloud environments. However, since I will be using these methods on edge devices, their inclusion here is particularly relevant to highlight their applicability in this context.

3.4 Deep Learning Methods for Video Streams in AWS Cloud

3.4.1 AWS SageMaker AI

Amazon SageMaker AI is a comprehensive, fully managed service designed to streamline ML processes. It empowers data scientists and developers to efficiently build, train, and deploy ML models in a production-ready environment with ease. SageMaker AI integrates seamlessly with multiple development environments, providing a user-friendly interface to manage end-to-end ML workflows.

The platform eliminates the need to set up and manage your own servers, allowing individuals and organizations to focus on collaboratively developing ML workflows more quickly. SageMaker AI supports managed ML algorithms optimized for processing vast datasets in a distributed setup. It also offers the flexibility to use custom algorithms and frameworks, enabling tailored, distributed training configurations to meet specific workflow requirements. With just a few steps, models can be securely deployed into a scalable environment directly through the SageMaker AI console. [58]

3.4.2 AWS Rekognition

Amazon Rekognition is a cloud service that simplifies the integration of advanced computer vision capabilities into your applications. Powered by robust deep learning technology, it requires no prior machine learning expertise. The service offers an intuitive API to analyze images and videos stored in Amazon S3, enabling developers to quickly add features like object detection, text recognition, unsafe content filtering, and facial recognition.

With Rekognition's face recognition capabilities, users can detect, analyze, and

compare faces for various applications, such as identity verification, cataloging, people counting, and enhancing public safety. Built on scalable deep learning technology, Rekognition can process billions of images and videos daily. The service continually evolves, learning from new data and incorporating new labels and features, ensuring that its capabilities remain cutting-edge. [59]

3.4.3 Other Services

There are other services available, however, some are nearing their End of Life (EoL). For instance, Amazon Lookout for Vision, which appears to be a promising solution for anomaly detection, is scheduled to be discontinued on October 31, 2025. [60] There are also semi-ready-made solutions available, though they require dedicated hardware. For example, AWS Panorama necessitates the use of either a Lenovo ThinkEdge SE70 or an AWS Panorama Appliance. [61], [62] Amazon SageMaker Edge Manager offered tools to manage machine learning models on edge devices, enabling optimization, security, monitoring, and maintenance across fleets of devices like smart cameras, robots, PCs, and mobile devices. However, this service has been discontinued as of April 26, 2024. [63]

4 Experiments

4.1 Existing Studies

Study by Muhammad [64] was able to achieves up to 99% bandwidth savings by utilizing edge computing together with cloud computing compared to the cloud-only configuration by filtering low-value data at the edge and sending only bounding box data for detected objects to the cloud. Study by Silva et al. [18] highlights that edge processing can reduce the data transmitted to the cloud by 98–99%, primarily by aggregating and filtering data locally at the edge. Study by Savaglio et al. [65] provides a comparative analysis of cloud-based and Edge Intelligence (EI)-based deployments for a video analytics task in an IIoT scenario, focusing on key performance metrics. The cloud-based deployment achieves a faster inference time of 0,01 seconds per frame, compared to 0,28 seconds on a Raspberry Pi 4 (or 0,07 seconds with a Google Coral Board for edge inference acceleration). However, the bandwidth usage is significantly higher in the cloud-based setup, requiring gigabytes to upload video streams, whereas the EI-based deployment only uses kilobytes for transmitting metadata or results. Additionally, the energy footprint of edge devices is much lower (e.g., 3,1W for data reading and 5,5W for inference on a Raspberry Pi 4) compared to the substantial power demands of cloud servers. Study by Yi et al. [66] shows similar results, that by using edge computing it is possible to achieve significant bandwidth savings.

The paper by Anjum et al. [67] presents a cloud-based video analytics framework optimized for scalable object detection and classification. Leveraging GPU-accelerated cloud resources, the framework achieves significant performance improvements, processing 175 GB of video data in 3 hours compared to 6,52 hours on CPU-only configurations. It supports multiple video formats and automates video stream analysis with minimal operator intervention. Case studies on vehicle and face detection demonstrated detection accuracies of 85% and 99,91%, respectively, highlighting the framework's robustness and efficiency. The study also identifies opportunities for integrating edge computing to minimize data transfer overhead and enable real-time analytics, paving the way for more energy-efficient and responsive video analysis solutions.

The RealEdgeStream (RES) framework by Ali et al. [68] achieves significant performance improvements in real-time video analytics by leveraging edge and cloud resources. Experimental results demonstrate that the ECM (Edge-Cloudlet-MultiResource) configuration processes 100 frames per second in real time, compared to only 15 fps for a cloud-only approach. This setup reduces task completion time by 49% and saves 99% of bandwidth by filtering low-value data at edge and in-transit nodes. These results highlight RES's ability to provide scalable, efficient, and low-latency analytics for large-scale video streams, outperforming traditional centralized systems.

In the study by Ali et al. [69], the edge-enhanced deep learning system achieves significant efficiency gains in large-scale video stream analytics by distributing tasks across edge, cloudlet, and cloud resources. Experimental results show that the Edge-Cloudlet-Cloud (ECC) configuration reduces processing time by 37% compared to a cloud-only setup, while the addition of filtration (ECCF) further improves efficiency, achieving a 71% gain by discarding low-value frames early in the pipeline. The ECCF setup processes 100,000 object recognition jobs in 194 minutes, compared to

666 minutes in cloud-only mode, highlighting its effectiveness in reducing latency and optimizing resource utilization for real-time analytics.

Table 4.1: Existing Studies Summary

Study	Bandwidth Savings	Inference Time (Edge vs. Cloud)	Processing Speed Improvement	Energy Efficiency	Other Benefits
Muhammad [64]	Up to 99%	N/A	N/A	N/A	Reduced data transmission by sending only bounding box data.
Silva et al. [18]	98–99%	N/A	N/A	N/A	Data aggregation and filtering at the edge.
Savaglio et al. [65]	Significant (KB vs. GB)	0,28s (RPi 4), 0,07s (Google Coral) vs. 0,01s (Cloud)	N/A	3,1W (data reading), 5,5W (inference on RPi 4) vs. high cloud power usage	Lower energy footprint, reduced data transmission.
Yi et al. [66]	Significant	N/A	N/A	N/A	Confirmed bandwidth savings through edge computing.
Anjum et al. [67]	Potential for savings	N/A	175GB video in 3h (GPU cloud) vs. 6,52h (CPU cloud)	N/A	Cloud-based analytics with edge potential for reducing data transfer overhead.
Ali et al. (RES) [68]	99%	N/A	100 fps (ECM) vs. 15 fps (Cloud)	49% task completion time reduction	Scalable, low-latency video analytics.
Ali et al. (ECCF) [69]	N/A	N/A	100,000 jobs in 194 min (ECCF) vs. 666 min (Cloud)	71% efficiency gain	Improved latency and resource optimization.

4.2 Deeper Look into the Research Questions

In this section, I will delve deeper into the research questions. I have included this chapter because, the research questions itself does not define for example, what I mean by "processing". For RQ1, I refer to the term "processing", which broadly encompasses any action that either the Raspberry Pi or AWS cloud must perform to complete the assigned task. Additionally, I address "cost" and "performance." Cost refers to the overall expenses involved in executing the provided tasks, while performance is a more nuanced concept. Performance may include factors such as latency, throughput, processing speed, bandwidth utilization, scalability, reliability, frame drop rate, or quality of analysis. For the purposes of this study, I have chosen to focus specifically on latency, processing speed, and bandwidth utilization. In the following, I will define these three performance metrics within the context of this thesis.

Latency: The time it takes from capturing the video frame until actionable results (e.g., detection or alerts) are available.

In an industrial setting, especially if real-time decision-making is crucial, latency is a major aspect of performance. As we have learned so far from the literature review, edge computing typically offers lower latency compared to cloud computing since processing occurs closer to the source, avoiding the delays associated with transmitting data to the cloud and back.

Processing Speed: The time required to perform computations, such as object detection or anomaly detection, on each video frame.

This refers to how quickly the system can run the analytics on the video data. On the edge, the available computational power is typically limited compared to the vast processing capabilities in the cloud, potentially leading to slower processing times for more complex tasks.

Bandwidth Utilization: The amount of network bandwidth required to trans-

mit data between the edge device and the cloud (e.g. 1 Mb/s).

This factor indirectly affects performance, particularly when cloud computing is involved. Streaming raw video data from the edge to the cloud requires considerable bandwidth, which can be a bottleneck, especially in environments with limited network capacity or unstable internet connections.

For RQ2, during the phase where I experiment with distributing processing tasks between the edge device and AWS cloud, I will explore which aspects of the processing can feasibly be distributed. Based on these findings, I will aim to determine the most optimal way to allocate tasks between the edge and the cloud.

4.3 Raspberry Pi 5 as a Edge Device

The Raspberry Pi is a compact, cost-effective, and multifunctional single-board computer created by the Raspberry Pi Foundation. While primarily designed to advance computer science education and innovation, it is also widely utilized by hobbyists, educators, and professionals for diverse projects. [70]

I am using Raspberry Pi 5 8GB as a edge device (Figure 4.1). Below (Table 4.2) is listed some specifications. First, I prepared the Raspberry Pi for use by setting it up. This involved attaching the active cooler (Figure 4.2), inserting the SD card, and installing the camera module (Figure 4.3). Afterward, I connected the mouse, keyboard, and display, followed by plugging in the power cable to complete the setup (Figure 4.4). Here you can see all the components that I am using (Table 4.3). Here you can see the system software stack (Table 4.4).

Table 4.2: Raspberry Pi 5 Specification Table

Specification	Raspberry Pi 5
CPU	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76
GPU	VideoCore VII
RAM	LPDDR4X-4267 SDRAM (8GB)
Display Output	Dual 4Kp60 HDMI with HDR support
Video Decoding	4Kp60 HEVC decoder
Wi-Fi	Dual-band 802.11ac
Bluetooth	Bluetooth 5.0 / Bluetooth Low Energy (BLE)
Ethernet	Gigabit Ethernet
Storage	microSD card slot
Camera Interfaces	2 × 4-lane MIPI camera transceivers
USB Ports	2 × USB 3.0 ports (5Gbps), 2 × USB 2.0 ports

Table 4.3: Raspberry Pi 5 Component Table

Component	Name
SD Card	64GB Micro SD Class A1
Cooling	Raspberry Pi 5 Active Cooler
Power Supply	Official Raspberry Pi 27W USB-C EU
Camera	Raspberry Pi Camera Module 3
Camera Accessories	Raspberry Pi Camera Cable 200mm
Other	Official Raspberry Pi Micro HDMI Cable 1m

Table 4.4: System Software Stack

Software	Version
Operating System	Raspbian GNU/Linux 12 (bookworm)
Kernel	Linux 6.6.51+rpt-rpi-v8
Architecture	arm64

Figure 4.1: Raspberry Pi 5 8GB.

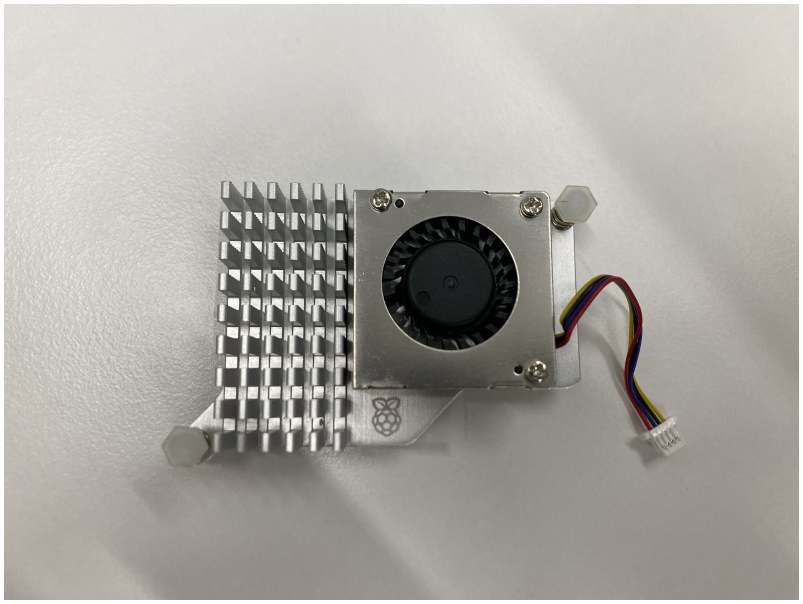

Figure 4.2: Active Cooler for Raspberry Pi 5.

Figure 4.3: Camera Module 3 for Raspberry Pi.

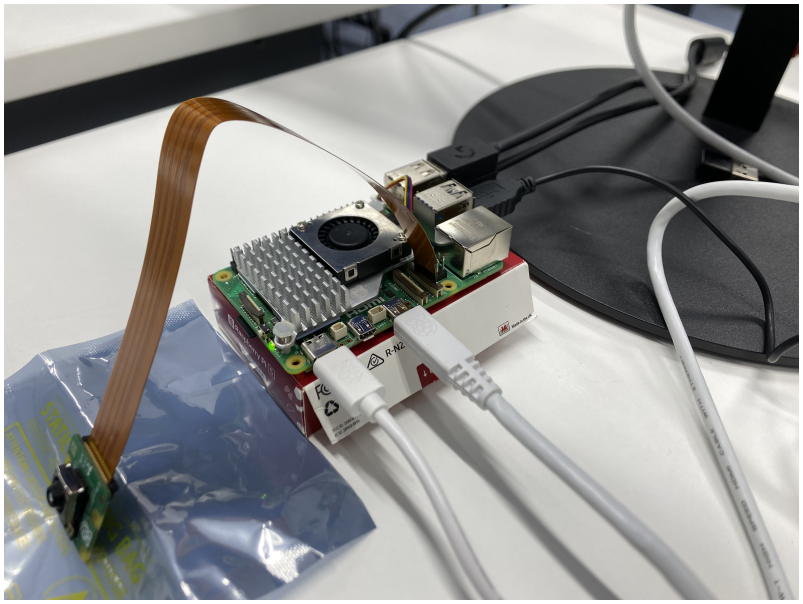

Figure 4.4: First Setup With Every Needed Component.

4.4 Experiment Setup

I will be experimenting with scenarios relevant to an IIoT environment, focusing on two key areas: basic live video streaming and object detection. The primary context for these scenarios is a conveyor belt and the items moving on it. Since I do not have access to a physical conveyor belt, I will simulate the setup by playing a conveyor belt video on a display and using the camera module to record the screen. I developed a Python script using the Pygame library to simulate a conveyor belt (Figure 4.5). The items moving along the conveyor belt are cars and bicycles, as these are included in the COCO label set. The experiments will be conducted in three phases: initially, processing will be performed exclusively on the edge device; next, it will be handled entirely on the AWS cloud; and finally, computational tasks will be distributed between the edge device and the AWS cloud. For the two key areas of interest, I will do the following five experiments.

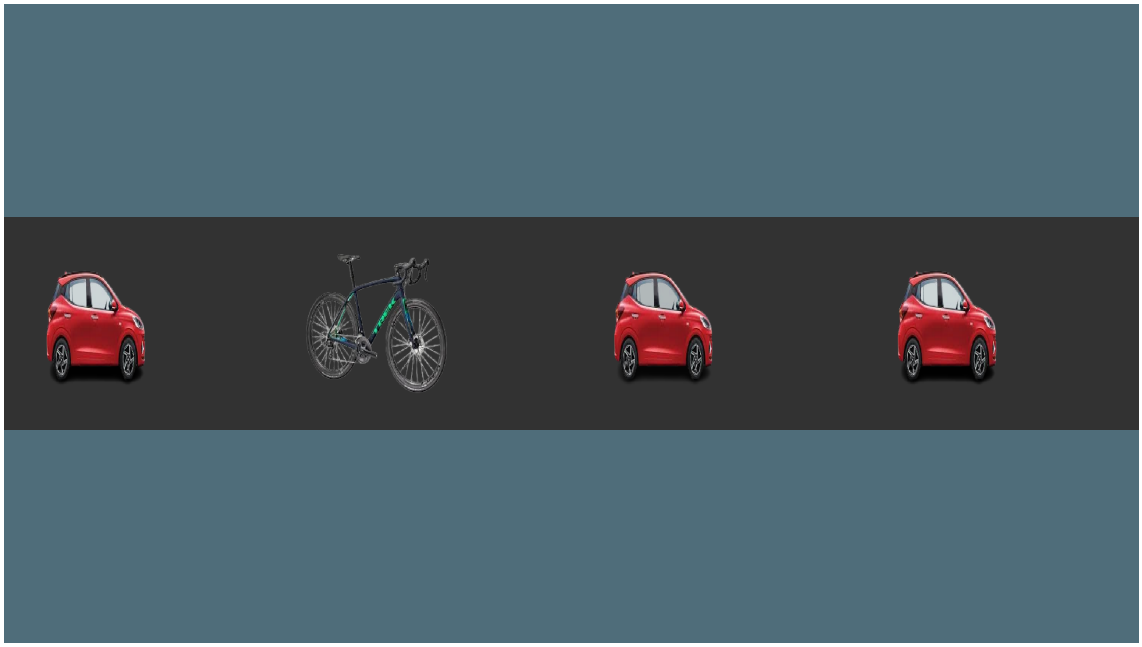

Figure 4.5: Conveyer Belt Animation.

It is important to note that since the Raspberry Pi 5 does not support hardware-accelerated video encoding, all encoding is performed in software, which may impact performance. Depending on the chosen encoding format, this can lead to increased CPU usage and latency. For instance, software-based H.264 encoding may introduce delays compared to hardware-accelerated alternatives found in earlier Raspberry Pi models. Additionally, the simulated conveyor belt animation does not include real-world video noise or compression artifacts, meaning that the results may differ from real industrial video streams where encoding artifacts, such as motion blur or blockiness, could affect object detection performance.

Experiment 1: Performance Evaluation of Live Video Streaming from Raspberry Pi 5 to Client Device Over Local Wireless Network Connection

This experiment is relatively straightforward. The Raspberry Pi 5 will be configured to capture and stream video in real-time over a local wireless network with an alleged

maximum speed of 600 Mb/s to a client device, which in this case is a MacBook Pro laptop. The primary focus of the experiment is to measure latency, specifically, the time it takes for the video stream to travel across the network to the client. The video streaming will be tested using three different protocols: User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Real-Time Streaming Protocol (RTSP).

To measure latency in this setup, the Raspberry Pi 5 is configured to capture video from a display showing a clock. The captured video is then streamed to a client device, where it is displayed on another screen. A smartphone camera is used to record both the original clock display, and the client display simultaneously. By analyzing the recorded video, it becomes possible to determine the latency by comparing the time shown on the original clock with the time displayed in the streamed video. This provides a visual representation of the delay between the source and the streamed output (Figure 4.6).

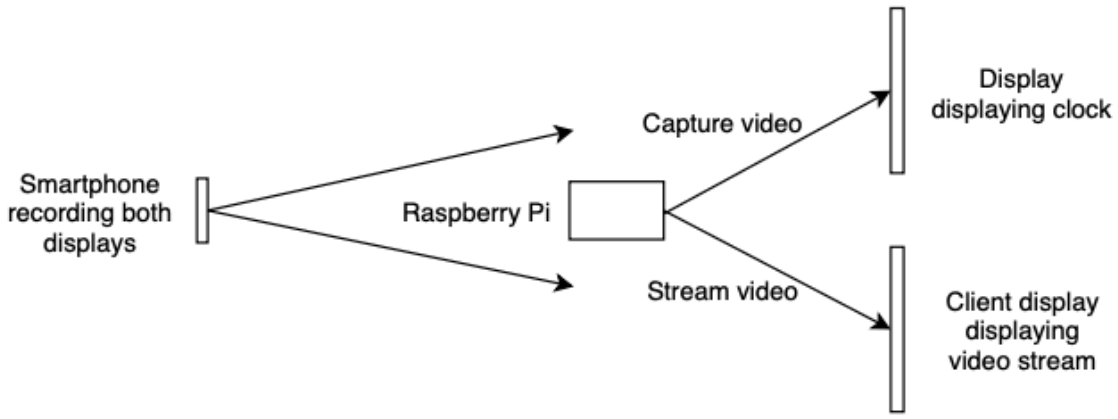

Figure 4.6: Latency Measurement Setup V1

This provides partial answers to RQ1, because it measures the edge performance.

The need for remote monitoring, such as overseeing a production process, is

highly relevant in the context of IIoT.

Experiment 2: Performance Evaluation of Object Detection on Raspberry Pi 5 with Local Wireless Network Communication

In this experiment, we focus exclusively on the Raspberry Pi 5, which performs object detection and transmits the results to a client device, a MacBook Pro laptop, over a TCP connection. The object detection is tested using two setups: the first employs Python code integrating OpenCV with the MobileNet V2 model and the COCO label set. Both setups include embedded logic to measure and analyze key performance metrics such as latency, processing time, and bandwidth utilization. The second setup uses Python code combining OpenCV with the YOLO V5 model and the COCO label set.

This script enables real-time object detection on a Raspberry Pi using TensorFlow Lite and the Picamera2 library, while simultaneously logging performance metrics and sending results over a TCP connection. It captures camera frames, preprocesses them for the model, and runs inference to detect objects, drawing bounding boxes and labels on the camera preview. Key metrics, including latency, processing time, and bandwidth utilization, are logged to a CSV file for performance analysis. Detection results, such as object labels, confidence scores, and bounding box coordinates, are transmitted to a client device via a configurable TCP connection.

This provides partial answers to RQ1 and RQ2, because it measures the edge performance.

Experiment 3: Live Video Streaming from Raspberry Pi 5 to AWS Cloud

This experiment is expected to be relatively straightforward. I will configure the Raspberry Pi 5 to capture and stream live video to the AWS cloud using the AWS producer library. The primary focus of the experiment is to measure latency, specif-

ically, the time required to transmit live video from the Raspberry Pi 5 to the AWS cloud. The experiment will involve comparing two different AWS regions: us-east-1 (eastern United States) and eu-central-1 (central Europe). This means that in one scenario, the video will be streamed to the eastern United States, while in the other, it will be streamed to central Europe.

The latency in this setup is measured using a method similar to that employed in experiment 1. The Raspberry Pi 5 is configured to capture video from a display showing a clock and stream the captured video to the AWS cloud. The streamed video is then accessed and played through the AWS management console, displayed on a separate screen. A smartphone camera is used to simultaneously record both the original clock display and the streamed video on the client display. By analyzing the recorded footage, the latency is determined by comparing the time shown on the original clock with the corresponding time displayed in the streamed video.

This experiment provides partial answers to all the research questions, as it provides cloud side statistics on latency, bandwidth utilization and the cost of streaming.

Experiment 4: Live Video Streaming from Raspberry Pi 5 to AWS for Cloud Processing with Result Feedback to Client

In this setup, the Raspberry Pi 5 will stream live video to AWS cloud, where object detection will be executed. Once the processing is complete, the AWS platform will send the results as string-based detection information back to the client device. This setup allows the client to receive and display processed detection results while utilizing the cloud for heavy computational tasks.

In this experiment, I will explore two methods for processing video stream using AWS cloud services. Two key services that can be used for this scenario are AWS SageMaker AI and AWS Rekognition. The accompanying diagram illustrates a

simplified infrastructure setup (Figure 4.7). The video stream originates from an on-premises Raspberry Pi camera and is streamed to AWS Kinesis Video Stream. From there, the stream can be directly forwarded to AWS Rekognition for processing. However, for AWS SageMaker AI, direct forwarding from Kinesis Video Stream is not supported. Instead, the video stream must first pass through an AWS Lambda function, and the Lambda function must be separately triggered in order for it to fetch the video stream, as Kinesis Video Stream itself cannot act as a trigger for a Lambda function. Both AWS Rekognition and AWS SageMaker AI can identify points of interest, which can then be stored in an S3 bucket. Additionally, the processed data can be sent to AWS SNS, allowing information to be relayed to the client either directly or via AWS SQS.

This experiment provides partial answers to all the research questions, as it provides cloud side statistics on latency, bandwidth utilization, processing speed and the cost of the used services.

I will begin with AWS Rekognition. According to the documentation, this service is well-suited for video streaming applications and offers capabilities such as identity verification, facial analysis, and object and activity recognition. For home monitoring scenarios, it can be utilized for tasks like recognizing pets or detecting delivered packages. [71] Before using AWS Rekognition, you must first configure a stream processor, the stream processor works under the hood of AWS Rekognition. However, when detecting objects, the video duration is limited to a maximum of 120 seconds. This limitation means creating a continuous video processing pipeline is not straightforward, it would necessitate chaining multiple stream processors to handle longer durations seamlessly.

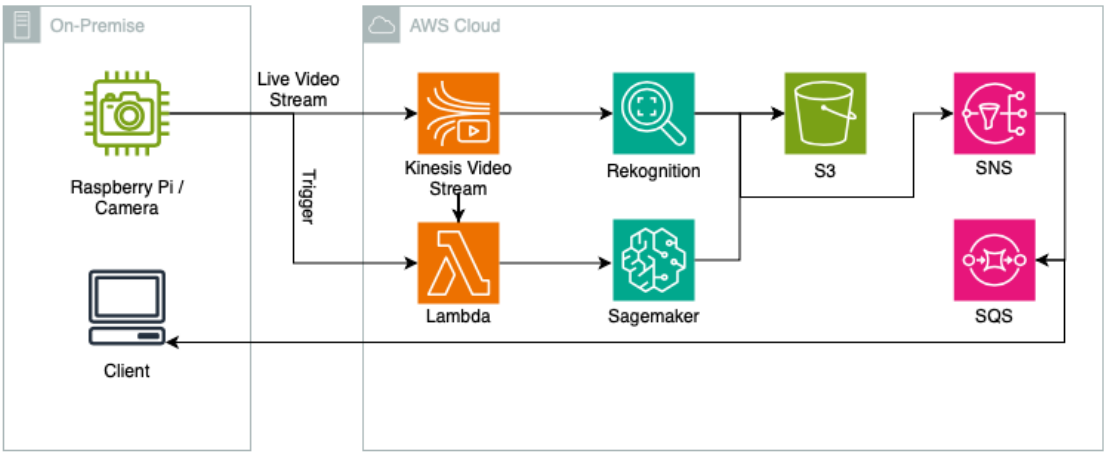

Figure 4.7: Simple AWS Infrastructure for Object Detection with Text-based Client Feedback

As I began setting up the environment for AWS SageMaker AI, I realized that a simpler approach would be more effective. While it is certainly possible to use SageMaker AI as described earlier, it is not necessary for my specific use case.

In this experiment, I have configured SageMaker AI as follows (Figure 4.8): The Raspberry Pi 5 captures video while simultaneously running a Python script that extracts a frame from the video feed, for example every second. This frame is then sent to AWS SageMaker AI in the cloud, where object detection is performed. The results are subsequently transmitted back to the client.

Additionally, the script includes logic to measure key performance metrics such as latency, processing speed, and bandwidth utilization. These metrics are recorded and saved in a CSV file for further analysis.

On the AWS cloud side, from the Amazon SageMaker AI management console's JumpStart menu, I selected the TensorFlow object detection model, which by default is SSD-ResNet50-v1. After choosing the model, I deployed it using the default instance type (ml.p3.2xlarge), and it was ready for use. The slowest instance type

supported for this model is ml.m5.large.

Once deployed, the model provides an endpoint that can be used to perform object detection on images or video frames. This default model comes pre-trained with the COCO 2017 dataset, enabling it to recognize a wide range of objects.

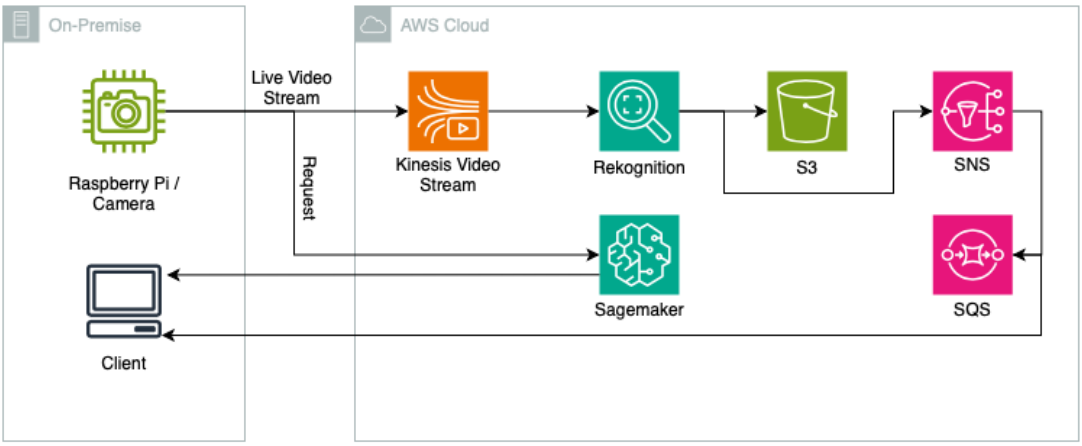

Figure 4.8: Simplified AWS Infrastructure for SageMaker

Experminent 5: Preprocessing Live Video on Raspberry Pi 5 before Streaming to AWS for Object Detection and Client Feedback

This experiment focuses on preprocessing a live video feed on the Raspberry Pi 5 before streaming it to the AWS cloud. Based on my findings throughout this study, AWS SageMaker AI is the most suitable service for this context, and it will be used for this experiment. Preprocessing at the edge may involve tasks such as reducing resolution, adjusting frame rates, or applying different encoding strategies. Specifically, this experiment aims to analyze how various frame manipulation techniques impact object detection performance, including modifications to resolution and frame size. Once preprocessed, the video frames will be sent to SageMaker AI for object detection, and the results will be returned to the client device as text-

based information. The entire process is managed through a Python script, which also includes logic to measure key performance metrics.

In this experiment I will also try the cheapest instance type (ml.m5.large) that is available for the SSD-ResNet50-v1 model, same model that was used on the experiment 4.

This experiment provides partial answers to all of the research questions, as it provides cloud side statistics on latency, bandwidth utilization, processing speed and the cost of the used services after the preprocessing has been done on the edge device. After combining the results of the experiments and findings from the literature review, we will have quite good understanding and answers for the research questions.

4.5 Challenges Encountered Prior to Experimentation

I have dedicated tens of hours to overcoming the technical challenges associated with these experiments. This was partly due to the fact that much of the material covered in this thesis was entirely unfamiliar to me before I began studying these topics. For instance, I had no prior experience with edge hardware, machine learning, or video streaming.

Amazon Kinesis Video Streams CPP Producer, GStreamer Plugin

The Amazon Kinesis Video Streams Producer SDK for C/C++ simplifies the process of developing on-device applications that securely connect to a video stream and reliably transmit video and other media data to Kinesis Video Streams. It manages essential tasks such as packaging frames and fragments from the device's media pipeline. Additionally, the SDK automates stream creation, token rotation to ensure

secure and continuous streaming, handling acknowledgments from Kinesis Video Streams, and other related processes. [72]

GStreamer is a versatile library designed for building graphs of media-handling components. It supports a wide range of applications, from basic Ogg/Vorbis playback and audio/video streaming to advanced tasks such as audio mixing and non-linear video editing. [73]

Most of the AWS documentation on video streams recommends using this library, so I decided to follow that guidance. However, I encountered a significant issue: the Amazon Kinesis Video Streams Producer Library did not function properly on the newest Raspberry Pi 5 model I was using. The primary reason for this was the lack of hardware-accelerated video encoding support on the Raspberry Pi 5. Without delving into the technical details, resolving these issues required nearly two weeks of discussions with one of the library's maintainers. Ultimately, we were able to get it working on the Raspberry Pi 5 by building and compiling the library with specific commands and running it using a precise configuration. What I took away from this experience is that even large organizations like AWS can have notable inaccuracies in their documentation. The AWS documentation claimed that the library would function as described on Raspberry Pi devices, including the Raspberry Pi 5 (stating compatibility with "Raspberry Pi 3 or later"). However, this was not the case. Following my complaints, the documentation was updated about three weeks later but I have not tested the new version.

Streaming Video Over a Network

I tested video streaming over a network from the Raspberry Pi to a client laptop using three different protocols, including UDP, TCP, and RTSP. Surprisingly, only the UDP and TCP protocols successfully transmitted the video stream, while the RTSP failed to function as expected. The underlying reason for this issue remains

unclear at this stage. Several potential causes could be contributing to this problem, such as firewall restrictions on either the Raspberry Pi or the client laptop, network configurations that block or deprioritize RTSP traffic, router settings or NAT policies interfering with non-TCP/UDP protocols, or even software misconfiguration or bugs in the streaming applications.

AWS IAM Permissions and Policies

Nearly every aspect of these experiments involving AWS required working with AWS Identity and Access Management (IAM) permissions and policies. Each service demands specific permissions to interact with other services and perform various actions. While I won't delve deeply into the details of IAM configurations in this thesis, I can note that, despite the AWS management console being relatively intuitive once you become familiar with it, unexpected issues can still arise, and I encountered some of those challenges during my work.

5 Result Analysis

5.1 Experiment 1: On-Premise Video Streaming Latency

The latency measurements were determined by analyzing a smartphone recording of the original clock display and the streamed video displayed on the client device. For UDP, the latency was observed to be around 2.5 seconds, a result that deviates from its typical reputation as a low-latency protocol. In contrast, TCP demonstrated a much lower latency of less than half a second, performing exceptionally well in this controlled experiment.

In some cases, TCP can achieve better throughput than UDP, especially when sending many small packets relative to the Maximum Transmission Unit MTU size. For example, an experiment showed that transmitting 300-byte packets over ethernet (with a 1500-byte MTU) resulted in TCP being 50% faster than UDP. This is because TCP buffers data to fill a full network segment before sending, making more efficient use of bandwidth. In contrast, UDP immediately transmits each small packet, leading to network congestion due to the high number of individual packets. [74] This congestion can also increase latency for UDP, as network devices may struggle to process and queue the flood of small packets efficiently.

These findings contribute to answering RQ1, as they provide a direct evaluation of edge computing performance in processing industrial video streams. By measuring

latency in a local network setup, this experiment demonstrates the capability of an edge device, specifically, the Raspberry Pi 5, to handle real-time video streaming with different protocols. The results show that edge computing can achieve low latency, particularly with TCP, but also highlight potential inefficiencies depending on protocol choice. However, since this experiment does not assess cloud-based processing, it only partially answers RQ1, offering insight into edge performance without a comparative analysis of cloud alternatives. This baseline is essential for later evaluating how cloud computing might differ in terms of cost and performance, completing the broader picture of the trade-offs between edge and cloud solutions.

5.2 Experiment 2: Object Detection Performance on Edge

The first setup, which utilized MobileNet V2, demonstrated an average latency of 49,81 milliseconds (Figure 5.1). The average processing time per frame was measured at 19,52 milliseconds (Figure 5.2), with bandwidth utilization recorded at 1,61 KB/s (Figure 5.3). These results indicate a relatively fast and efficient object detection pipeline, highlighting MobileNet V2's ability to perform lightweight inference on resource-constrained devices like the Raspberry Pi 5.

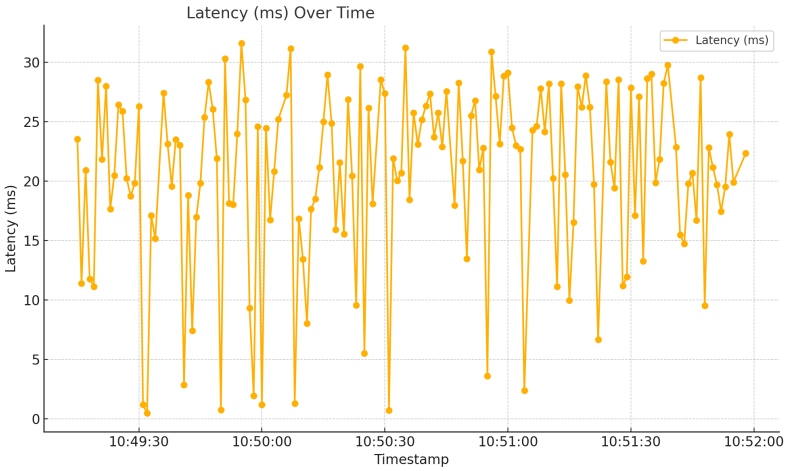

Figure 5.1: Processing on Raspberry Pi (MobileNet V2): Latency (ms)

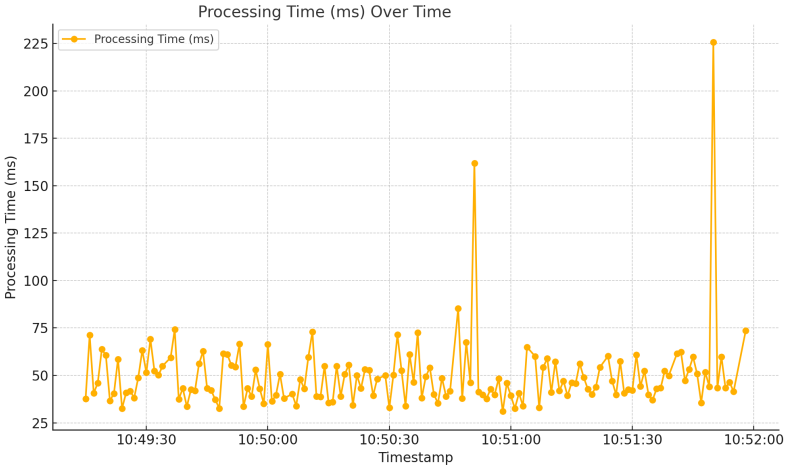

Figure 5.2: Processing on Raspberry Pi (MobileNet V2): Processing Speed (ms)

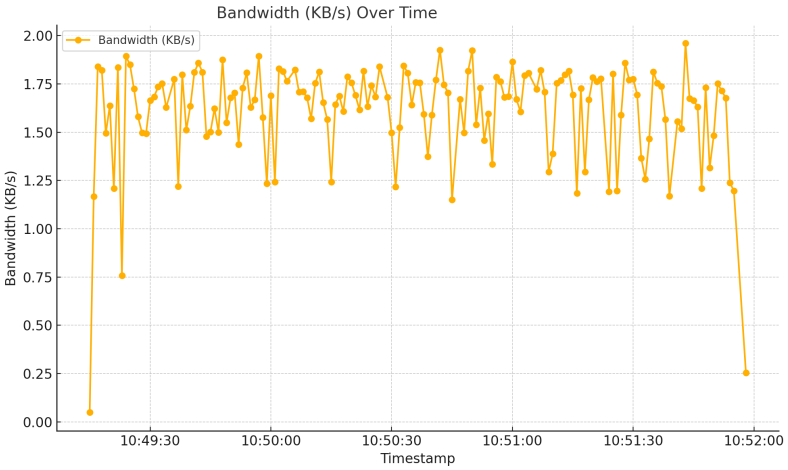

Figure 5.3: Processing on Raspberry Pi (MobileNet V2): Bandwidth Utilization (KB/s)

The second setup, employing YOLO V5, achieved a significantly lower average latency of 12,43 milliseconds (Figure 5.4). However, this came at the expense of a much longer average processing time of 478,26 milliseconds per frame (Figure 5.5). Bandwidth utilization was slightly higher, at 1,77 KB/s (Figure 5.6). These results suggest that while YOLO V5 is capable of detecting objects more quickly in terms of communication latency, its computational demands significantly increase the time required for processing each frame.

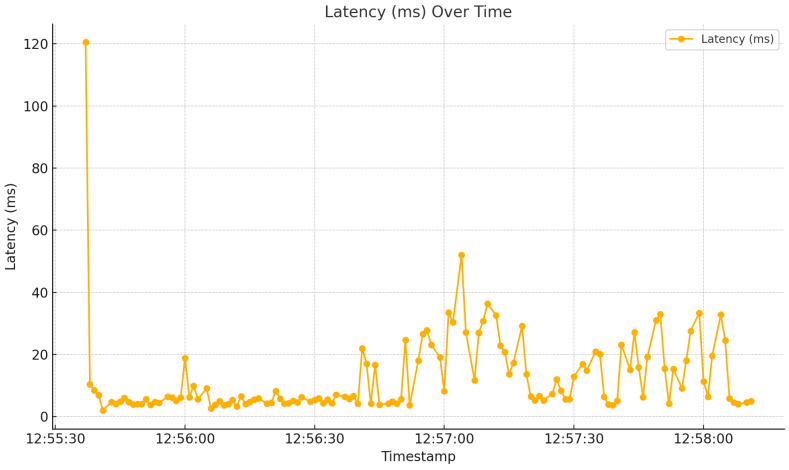

Figure 5.4: Processing on Raspberry Pi (YOLO V5): Latency (ms)

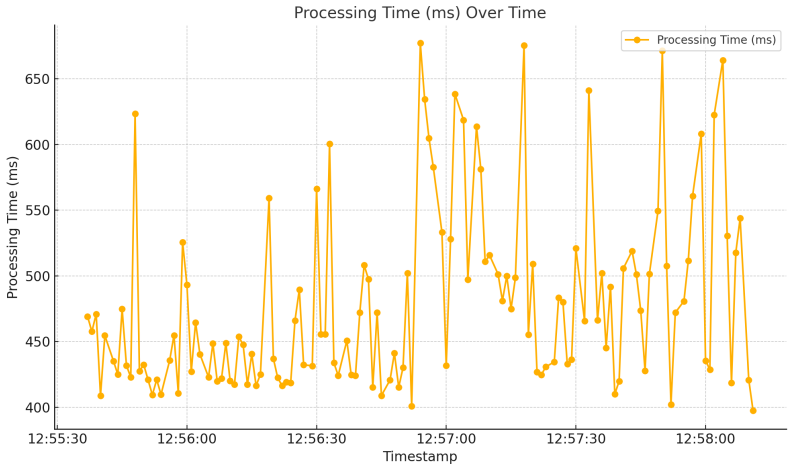

Figure 5.5: Processing on Raspberry Pi (YOLO V5): Processing Speed (ms)

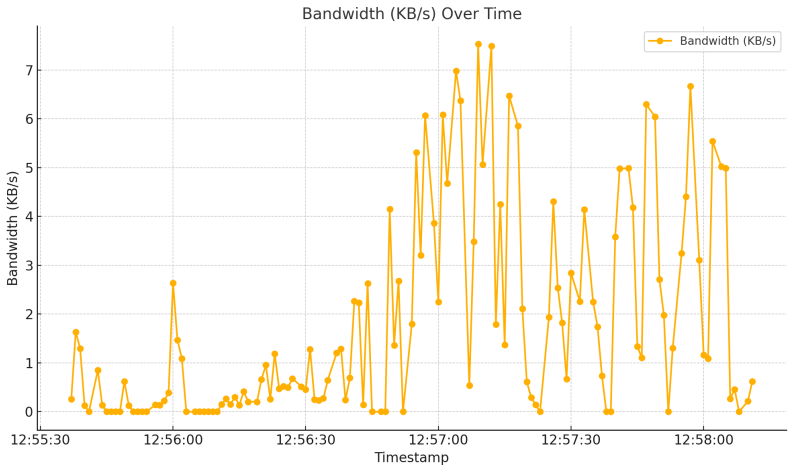

Figure 5.6: Processing on Raspberry Pi (YOLO V5): Bandwidth Utilization (KB/s)

The performance comparison between MobileNet V2 and YOLO V5 reveals trade-offs between computational efficiency and detection accuracy. MobileNet V2 demonstrated significantly faster processing times, with each frame requiring an average of 19,52 milliseconds to be analyzed. This performance aligns with the model's lightweight design, optimized for real-time applications on devices with limited computational resources. The higher latency observed in this setup (49,81 milliseconds) may be attributed to the additional time required to transmit results over the TCP connection, which could include buffering and preprocessing overheads.

In contrast, YOLO V5 exhibited a much lower average latency of 12,43 milliseconds, suggesting that its results were transmitted more quickly to the client device. However, the average processing time per frame, at 478,26 milliseconds, indicates a much heavier computational burden. YOLO V5 is a more complex and resource-intensive model, designed to deliver high detection accuracy. This complexity likely caused the Raspberry Pi 5 to take longer to analyze each frame, resulting in a significant delay at the processing stage. The slightly higher bandwidth utilization of 1,77 KB/s for YOLO V5, compared to 1,61 KB/s for MobileNet V2, can be explained by the increased data volume generated from its more detailed and potentially more frequent object detection results.

These findings provide more partial answers to RQ1, specifically on the edge computing side, by demonstrating how different object detection models perform on a resource-constrained device like the Raspberry Pi 5. The experiment highlights key trade-offs in processing time, latency, and bandwidth utilization, offering valuable insight into the efficiency of edge computing for industrial video streams. However, to fully answer RQ1, it is necessary to extend the experiments to cloud-based processing. Comparing these edge results with cloud performance and cost will provide a more comprehensive understanding of the trade-offs between edge and cloud computing, ultimately leading to a complete answer to the research question.

5.3 Experiment 3: Cloud Video Streaming Latency

The observed latency for both AWS regions was consistent at around 7 seconds. Whether the video was streamed to us-east-1 in the eastern United States or eu-central-1 in central Europe, the delay remained comparable. This outcome is noteworthy given the geographic proximity of central Europe to Finland, which might have been expected to result in a noticeably lower latency compared to the eastern United States.

It is also important to consider that the latency measurements included both upstream transmission from the Raspberry Pi 5 to the AWS cloud and downstream processing to display the video on the AWS management console. Thus, the overall latency reflects not only network transmission times but also the time required for video encoding, buffering, and rendering at various stages. The latency may be lower or higher in other scenarios, such as when a service processes or utilizes the video.

The bandwidth utilization during the experiment, as observed in the CloudWatch PutMedia Incoming Data graph, was consistently around 4.9 Mb/s (Figure 5.7). This indicates that the upstream video feed from the Raspberry Pi 5 to AWS Kinesis Video Streams was stable and maintained a steady bitrate.

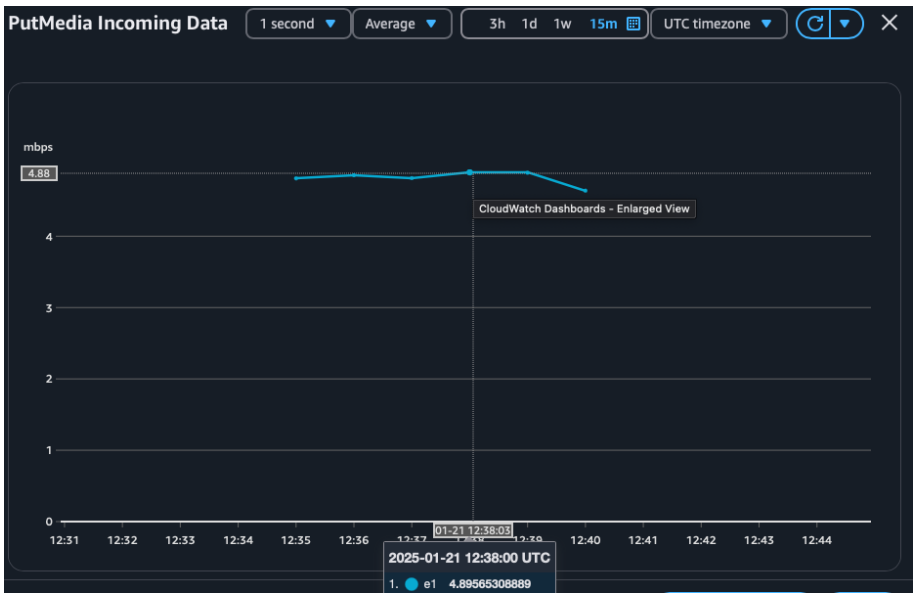

Figure 5.7: AWS Kinesis Video Stream Putmedia Incoming Data Average (Mb/s)

For real-time applications or latency-sensitive scenarios, these findings highlight the need to evaluate the entire system architecture holistically. While network distance plays a role, factors such as video encoding, processing times, and resource allocation across the pipeline often have a more significant impact on latency.

These findings provide partial answers to RQ1, now expanding the analysis to

cloud-based processing and getting closer to a comprehensive comparison between edge and cloud computing. The experiment demonstrates that streaming video to the AWS cloud introduces significantly higher latency (around 7 seconds) compared to local edge streaming. This highlights the potential challenges of using cloud infrastructure for real-time industrial video applications, where lower latency is often a critical requirement. Additionally, by analyzing bandwidth utilization (4.9 Mb/s) and the factors contributing to cloud latency, such as encoding, buffering, and rendering, this experiment sheds light on the trade-offs involved in cloud-based video streaming. However, to fully answer RQ1, further experiments are needed to compare not only performance but also cost and potential optimizations in cloud processing. By adding these cloud-side insights, this experiment brings the research closer to a complete evaluation of the trade-offs between edge and cloud computing for industrial video streaming. This also provides partial answers to RQ 2 and RQ 3, since in this case we didn't preprocess the streamed video in any way.

5.4 Experiment 4: Object Detection Performance on Cloud

AWS Rekognition: Based on the timestamps extracted from the AWS SQS notification, the estimated end-to-end latency was approximately 7 seconds. This value was determined by comparing the kvsProducerTimestamp, which marks when the video frame was ingested into AWS Kinesis Video Streams, with the timestamp in the SNS notification, which represents when the processing results were published. The difference between these timestamps provided a concrete measure of how long it took from capturing the video frame to delivering actionable detection results.

The CloudWatch metrics for Kinesis Video Streams showed a consistent upstream data rate of approximately 4,9 Mb/s, indicating a stable transmission from

the Raspberry Pi 5 to AWS. This was somewhat to be expected based on the previous experiment.

Through this experiment, I learned that AWS Rekognition may not be the most suitable option for continuous video stream analysis. While it could be applied in an IIoT context to recognize employee faces, I believe it is not an ideal choice for tasks such as conveyor belt analysis. After this realization, I did not study the AWS Rekognition further.

AWS SageMaker AI: The experiment utilized an ml.p3.2xlarge instance to run the TensorFlow SSD-ResNet50-v1 object detection model. The average frame size processed was 0,072 MB, indicating that each frame sent to AWS SageMaker was relatively lightweight, optimizing bandwidth usage. The Raspberry Pi 5 extracted frames at an average interval of 1.807 seconds, ensuring a balance between data transmission efficiency and real-time processing feasibility. The object detection model took an average of 6,085 seconds to process each frame (Figure 5.9), which highlights the computational demands of running object detection on the chosen AWS instance. The overall end-to-end latency, including capture, transmission, processing, and response time, was measured at 7,892 seconds (Figure 5.8). While this latency is acceptable for non-time-critical applications, real-time video analytics would require further optimizations. The system utilized an average bandwidth of 0,014 MB/s (Figure 5.10), reflecting an efficient use of network resources given the experimental constraints.

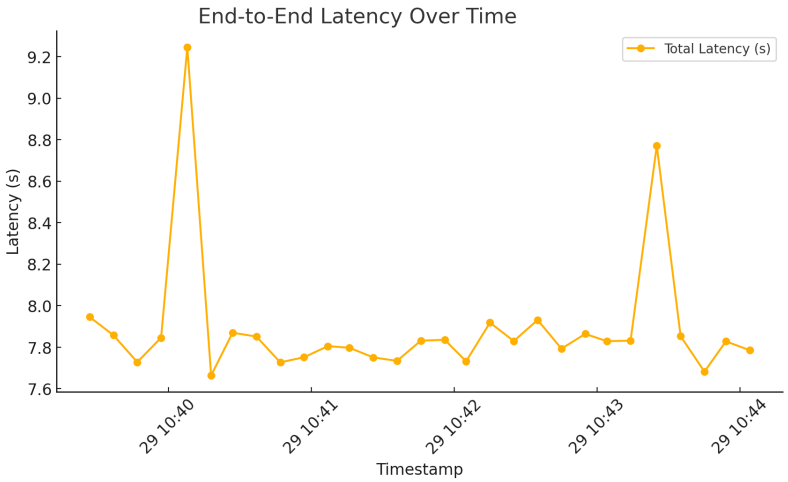

Figure 5.8: AWS SageMaker AI (ssd-resnet50-v1): Latency (ms)

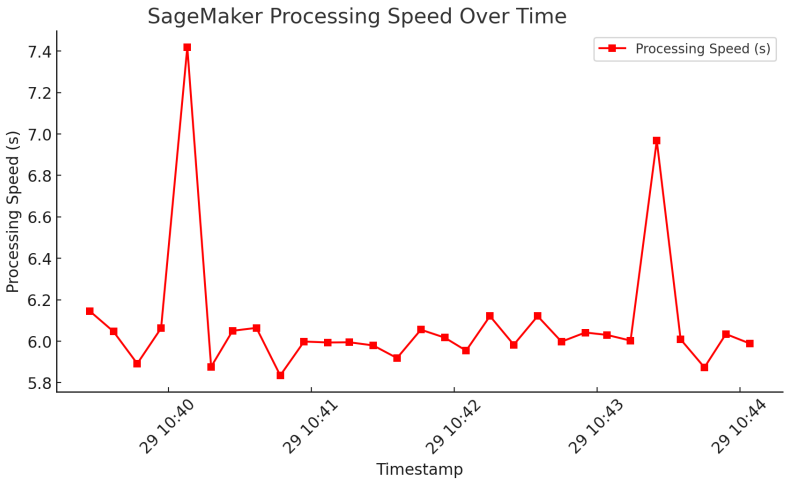

Figure 5.9: AWS SageMaker AI (ssd-resnet50-v1): Processing Speed (ms)

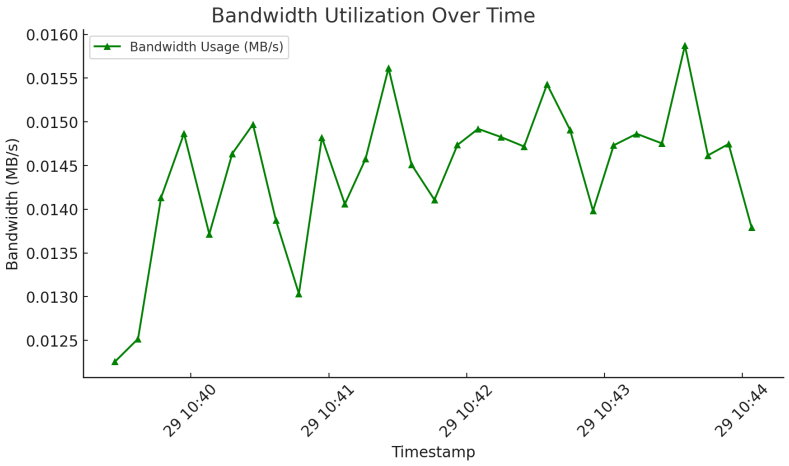

Figure 5.10: AWS SageMaker AI (ssd-resnet50-v1): Bandwidth Utilization (Mb/s)

The results suggest that AWS SageMaker AI provides a viable method for cloud-based object detection in live-streaming scenarios. However, certain limitations and considerations were observed. With an end-to-end latency of approximately 7.9 seconds per frame, the real-time nature of object detection is compromised. This delay stems primarily from the processing time on the SageMaker instance, making it less suitable for applications requiring instantaneous results such as autonomous navigation or security alert systems. The ml.p3.2xlarge instance was used by default, but the slowest instance capable of running the model is ml.m5.large. While a downgrade to a more cost-effective instance could reduce expenses, it would likely further increase processing time and latency. The decision to extract frames at one-second intervals ensured a manageable data load but might not be optimal for use cases requiring continuous video analysis. Reducing this interval, for example capturing frames every 500ms, could provide more responsive detections at the cost of higher processing demands. Unlike AWS Rekognition, which imposes a strict 120-second limit per video segment, AWS SageMaker AI allows for greater flexibility in processing frames. However, the need to manually handle frame extraction and transmission adds complexity to the system design.

This experiment further complements RQ1 by extending the evaluation of cloud computing for industrial video stream processing, providing crucial insights into performance, cost, and bandwidth utilization. The results highlight the trade-offs of cloud-based object detection, particularly the high latency (7.9 seconds per frame) observed with AWS SageMaker AI. While the cloud can handle computationally intensive tasks like object detection, the delay introduced makes it unsuitable for real-time applications, reinforcing the need for strategic decision-making when choosing between edge and cloud solutions. Additionally, the bandwidth efficiency of 0.014 MB/s suggests that cloud-based frame processing minimizes network load compared to full video streaming, which can be advantageous in constrained network environments.

Beyond RQ1, this experiment partially answers RQ2, as it provides evidence on what types of video stream processing tasks are better suited for cloud computing. The results indicate that AWS Rekognition, while effective for short, predefined video segments, is not ideal for continuous industrial video analysis. On the other hand, AWS SageMaker AI offers more flexibility but introduces significant latency, making it more suitable for tasks that do not require immediate results. This suggests that real-time decision-making tasks should remain at the edge, while more complex, non-time-sensitive analytics can be offloaded to the cloud to balance performance and computational cost.

Additionally, the findings partially address RQ3, as they demonstrate the impact of preprocessing video at the edge on communication overhead. Instead of streaming the entire video feed, the experiment used frame extraction at one-second intervals, significantly reducing bandwidth usage compared to continuous video transmission (from 4.9 Mb/s in experiment 3 to 0.014 MB/s in this setup). This supports the idea that preprocessing at the edge, such as extracting key frames instead of streaming raw video, can dramatically lower network demands while still enabling meaning-

ful cloud-based processing. However, further exploration is needed to determine the optimal balance between frame rate selection, detection accuracy, and network efficiency for different industrial applications.

5.5 Experiment 5: Hybrid Approach for Object Detection

The most significant finding from this experiment was the impact of the AWS SageMaker AI instance type on object detection performance. As anticipated, the ml.m5.large instance performed noticeably worse than the default instance used in experiment 4. The processing speed was slower, it averaged on 7,18 seconds (Figure 5.12) leading to increased total latency, which averaged on 8,97 seconds (Figure 5.11).

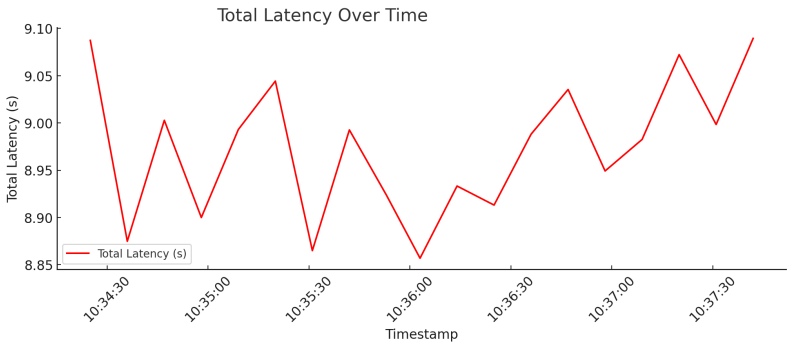

Figure 5.11: AWS SageMaker AI (ssd-resnet50-v1), Cheapest Instance Type (ml.m5.large): Latency (ms)

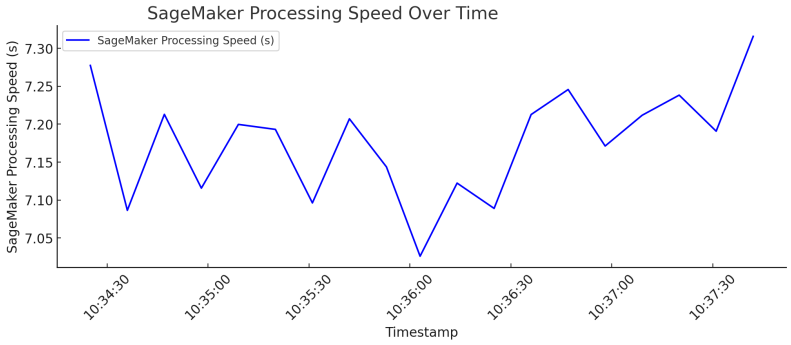

Figure 5.12: AWS SageMaker AI (ssd-resnet50-v1), Cheapest Instance Type (ml.m5.large): Processing Speed (ms)

One of the key focuses of this experiment was to determine whether preprocessing video frames on the edge device (Raspberry Pi 5) could enhance object detection efficiency. The preprocessing techniques tested included reducing frame resolution, adjusting frame size and reading images from memory instead of disk. Surprisingly, these modifications did not yield significant performance improvements. Despite efforts to reduce transmission time, the overall system latency and detection accuracy remained largely unaffected. This suggests that the primary bottleneck in this setup lies in the inference process on AWS SageMaker AI rather than in the preprocessing stage at the edge.

This experiment further complements RQ1 by refining the understanding of trade-offs between edge and cloud computing for industrial video stream processing, particularly in terms of cost and performance. By utilizing the cheapest available AWS SageMaker AI instance (ml.m5.large), the study directly evaluates the impact of cloud computational power on processing speed and latency. The results indicate that reducing cloud costs comes at the expense of increased processing time, with the average processing speed dropping to 7,18 seconds per frame and total latency rising to 8,97 seconds. This confirms that cost and performance are closely linked in cloud computing, reinforcing the importance of selecting the right balance between

cloud resource allocation and real-time processing needs.

This experiment also partially answers RQ2 by investigating the role of preprocessing at the edge and its potential to optimize the distribution of processing tasks between the edge and cloud. The findings reveal that reducing resolution, adjusting frame size, and optimizing memory usage on the Raspberry Pi 5 did not significantly improve object detection efficiency. This suggests that preprocessing alone is not enough to mitigate cloud latency, and that certain computationally heavy tasks, such as object detection inference, remain inherently cloud-dependent. These results imply that while preprocessing may help in reducing data transmission loads, it does not eliminate the need for powerful cloud computing resources when handling complex industrial video analysis.

Additionally, this experiment provides further insights into RQ3, particularly regarding the impact of edge preprocessing on communication overhead. While preprocessing techniques aimed at reducing data size were tested, the fact that they did not lead to substantial performance gains suggests that in this specific case, network transmission was not the primary bottleneck. Instead, the cloud inference time was the limiting factor, meaning that reducing the volume of transmitted data does not necessarily equate to lower overall latency. This highlights that while preprocessing at the edge can improve efficiency in some cases, its effectiveness depends on the nature of the processing task and the system's overall architecture.

5.6 Cost Analysis for Different AWS Services

Table 5.1: AWS Kinesis Video Stream Cost in Central Europe [75]

Action	Price
1 Gb of data ingested into Kinesis Video Streams	$0,01020
1 Gb of data consumed from Kinesis Video Streams	$0,01020

As observed in experiment 3, streaming video to AWS Kinesis Video Stream consumed an average of 4.9 Mb/s of bandwidth. Based on this, running the stream for 24 hours would result in a total bandwidth usage of 413.4 GB, costing approximately $4.21 per 24 hours (calculated at $0.01020/GB). While storage duration impacts the pricing of Kinesis Video Stream, a detailed analysis of these differences is beyond the scope of this thesis.

Table 5.2: AWS Rekognition Cost in Eastern US for Streaming Video Events [76]

Action	Price
Label Detection	$0,00817/min
Face Search	$0,12/min

For this analysis, I am using AWS Rekognition pricing from the Eastern US region, as label detection is not available in certain regions, such as Central Europe. Calculating the cost for continuous label detection over 24 hours, it would amount to $11.76 (at $0.00817/min). However, as discussed in this thesis, AWS Rekognition is better suited for analyzing motion events rather than continuous processing. For instance, if an event occurred once per hour and triggered AWS Rekognition only during that specific event, the cost would be significantly reduced.

Table 5.3: AWS SNS Notification Delivery Cost in Central Europe [77]

Action	Price
Email notification	$2,00/100000 notifications
HTTP/s notification	$0,60/million notifications

As we can see from the table (Table 5.3), AWS SNS is rather cheap. The cost of receiving notifications about an object being detected every five seconds over HTTPS for 24 hours using AWS SNS would be approximately $0.01037.

Table 5.4: AWS SQS Standard Queue Request Cost in Central Europe [78]

Action	Price
Zero-million requests/month	Free
Million-100 billion requests/month	$0,40/million notifications

As we can see from the table (Table 5.4), AWS SQS is also quite cheap. If we would poll for information every second for 30 days, we would end up making 2592000 requests and that would cost $1,04.

Table 5.5: AWS SageMaker AI Standard Instances Cost in Central Europe [79]

Instance type	vCPU	Memory	Price
ml.m5.large	2	8Gb	$0,138/hour

Table 5.6: AWS SageMaker AI Accelerated Computing Cost in Central Europe [79]

Instance type	vCPU	Memory	Price
ml.p3.2xlarge	8	61Gb	$4,779/hour
ml.p3.8xlarge	32	244Gb	$18,35/hour
ml.p4d.24xlarge	96	1152Gb	$47,0865/hour

The first instance listed in the table, ml.p3.2xlarge (Table 5.6), is the default choice when selecting the TensorFlow object detection model introduced earlier. Based on its hourly rate, running object detection continuously for 24 hours would cost $114,70. As evident from this calculation, even using the default instance type, running object detection 24/7 on AWS SageMaker AI would be quite expensive.

From the cost analysis presented, it is evident that using AWS services for real-time object detection in live video streaming scenarios within IIoT environments can be expensive. The continuous transmission of video data to AWS Kinesis Video Streams incurs significant bandwidth costs, while AWS Rekognition, when used for

continuous label detection, further amplifies the overall expense. Although services like AWS SNS and AWS SQS are relatively inexpensive, the real cost driver is the computational resource requirement for running object detection models, particularly when utilizing AWS SageMaker AI. The high hourly cost of GPU-based instances makes continuous processing expensive for large-scale deployments.

Given that edge hardware is often already deployed in IIoT settings for latency and reliability reasons, an optimized approach would be to shift object detection workloads from the cloud to on-premise edge devices. By running models locally on edge hardware, organizations can significantly reduce both bandwidth and compute costs while still leveraging the cloud for occasional processing, model updates, and storage when necessary. This hybrid approach ensures cost efficiency while maintaining the benefits of cloud scalability for specific use cases.

6 Discussion

This section reflects on the research questions posed in the thesis and examines how the findings contribute to the optimization of video streaming and analysis within IIoT environments. The study explored the balance between edge and cloud computing for IIoT video processing, analyzed how processing tasks should be allocated between the two, and assessed the impact of edge-based preprocessing on network communication.

Addressing the Research Questions

RQ1: What are the trade-offs between edge and cloud computing for processing industrial video streams in terms of cost and performance?

The results of the study, particularly from experiments 1 and 3, provide a clear comparison of streaming latency between edge and cloud computing. Experiment 1, where live video was streamed locally from the Raspberry Pi 5 over a wireless network to on-premise client, demonstrated an average latency of less than half a second. In contrast, experiment 3, where the Raspberry Pi 5 streamed video to the AWS cloud, exhibited a significantly higher latency of approximately 7 seconds. This substantial increase in delay underscores the performance limitations of cloud-based streaming, particularly for real-time IIoT applications where immediate feedback is crucial.

Experiments 2 and 4 further highlight the trade-offs in computational perfor-

mance between edge and cloud processing. In experiment 2, object detection was performed entirely on the Raspberry Pi 5 using MobileNet V2 and YOLO V5. The processing speed for MobileNet V2 averaged around 19,52ms per frame, while YOLO V5 required approximately 12,43ms per frame. On the other hand, experiment 4, which offloaded object detection to AWS SageMaker AI and AWS Rekognition, exhibited a much slower processing speed, completing inference in as high as 6 seconds per frame on a high-performance cloud instance. However, when including network transmission time and processing overhead, the total turnaround time increased to almost 8 seconds, making cloud processing impractical for low-latency applications.

The cost factor is another key consideration in evaluating the trade-offs between edge and cloud computing. The Raspberry Pi 5, which was used as the edge device, costs approximately $100 upfront, with no recurring operational costs aside from electricity consumption. In contrast, streaming live video to AWS Kinesis Video Streams incurs ongoing expenses. As we calculated in the cost analysis, it would cost approximately $4,21/24h to stream video to AWS through the Kinesis Video Stream service. The monthly cost would be approximately $125.

For cloud-based processing, AWS Rekognition label detection would cost approximately $350/month. AWS SageMaker AI, which we identified to be a better option would cost approximately $3440/month with the default instance type and around $100 with the cheapest instance type.

RQ2: What type of video stream processing tasks should be distributed between the edge and the cloud for optimal performance in industrial IoT systems?

A key insight from the existing studies reviewed in section 4.1 is that preprocessing tasks should primarily be performed at the edge, while more computationally intensive tasks should be handled in the cloud. Studies by Muhammad (2021) and Silva et al. (2020) demonstrated that edge-based filtering and preprocessing can re-

duce cloud-bound data volume by up to 99%, significantly lowering bandwidth costs and improving response times. Similarly, the RealEdgeStream (RES) framework by Ali et al. (2021) demonstrated a 49% reduction in task completion time when preprocessing was performed at the edge before sending processed data to the cloud for final inference.

The experiments conducted in this thesis further confirm these findings. Experiment 2 (edge-based object detection) showed that simple object recognition tasks could be efficiently performed on the Raspberry Pi 5, reducing the need for cloud processing. Experiment 5 (hybrid approach) demonstrated that a combined strategy, where the edge handles initial detection and filtering, while the cloud performs complex analysis and storage, is optimal for balancing cost, latency, and performance.

Based on both prior research and experimental findings, the optimal task distribution is as follows. Tasks best suited for the edge: motion detection and frame selection, object filtering (e.g., detecting if an object is present before sending data), initial classification using lightweight models like MobileNet V2 and low-latency applications where real-time response is critical. Tasks best suited for the cloud: high-accuracy classification using deep learning models, long-term storage and data analysis, large-scale anomaly detection requiring extensive computational power.

RQ3: How does video preprocessing at the edge impact communication overhead in industrial IoT systems?

One of the most significant impacts of edge-based preprocessing is the reduction of communication overhead, particularly in terms of bandwidth usage and cloud service costs. The cost analysis revealed that streaming unprocessed video to the cloud incurs high expenses, with AWS Kinesis Video Streams alone costing $100 per month per camera. By contrast, preprocessing at the edge reduces data transmission volume by up to 99%, significantly lowering operational expenses.

Experiments 4 highlight the benefits of preprocessing. In experiment 4, the edge device successfully detected objects and transmitted only relevant metadata, cutting down on unnecessary video transmission.

These findings strongly support a hybrid edge-cloud model, where preprocessing occurs at the edge to optimize communication efficiency while the cloud is leveraged for high-level data processing. This approach ensures a cost-effective, low-latency, and scalable solution for industrial IoT applications.

Reliability and Limitations

This study aimed to ensure reliability by maintaining a controlled experimental setup, using standardized measurement techniques, and repeating tests to verify consistency. Every effort was made to minimize variability, from carefully monitoring key parameters to collecting data under identical conditions. However, some limitations remain. The study was conducted using a single type of edge device, which means the findings might not apply to other hardware configurations. Since the experiments took place in a controlled setting, they may not fully reflect real-world industrial environments where factors like lighting changes or network fluctuations could impact performance. Another limitation is that the research focused solely on AWS, so there's no direct comparison with other cloud providers, which might offer different performance or cost benefits. Lastly, while the study is centered on industrial IoT applications, the findings may not directly translate to other fields like healthcare or smart cities. Despite these limitations, the results provide valuable insights, and future research can build on them by testing additional hardware, running real-world experiments, and exploring alternative cloud solutions.

7 Conclusion and Further Research

This thesis examined the trade-offs between edge and cloud computing for processing industrial video streams, focusing on cost, performance, and bandwidth utilization. Through a series of controlled experiments, the study evaluated how different processing distributions, edge-only, cloud-only, and hybrid approaches, affect key performance metrics such as latency, processing speed, and network efficiency in an IIoT environment. The findings show that while cloud computing provides robust processing capabilities, it introduces increased latency and higher bandwidth consumption, making it less suitable for real-time applications. In contrast, edge computing significantly reduces latency and limits the amount of data transmitted to the cloud, leading to noticeable bandwidth savings. Although existing studies have reported up to 99% reductions in data transmission through edge preprocessing, the experiments conducted in this thesis found similar trends, confirming that preprocessing at the edge effectively decreases network load and improves efficiency.

The research demonstrated that a hybrid approach, where essential preprocessing occurs at the edge while more computationally intensive tasks are handled in the cloud, offers the best balance between cost and performance. Simple preprocessing tasks such as object detection and motion filtering can be efficiently executed on the edge device, while detailed object classification and anomaly detection benefit from cloud-based processing power. This division allows for a significant reduction in bandwidth usage without compromising analytical accuracy. The experiments

further showed that preprocessing at the edge plays a crucial role in minimizing communication overhead. By filtering out unnecessary data before transmission, the overall volume of streamed information is reduced, leading to improved network efficiency and lower operational costs.

Overall, this study has demonstrated that a carefully structured combination of edge and cloud computing can enhance the efficiency of industrial video stream processing. While edge devices have hardware limitations that prevent them from handling all computational tasks independently, their ability to preprocess and filter data before cloud transmission proves highly beneficial. These findings contribute to the broader discussion on optimizing IIoT video analytics by highlighting the advantages of a hybrid computing model. By strategically distributing tasks between edge and cloud environments, IIoT systems can achieve real-time performance while maintaining cost-effectiveness and scalability.

Further Research

Further research could extend the findings of this study by exploring adaptive hybrid computing models, where the decision to process video at the edge or in the cloud is made dynamically based on real-time conditions. In experiment 5, preprocessing techniques were tested on the Raspberry Pi 5, but their impact on system performance was limited. Future work could investigate more advanced preprocessing strategies, such as intelligent frame selection based on motion detection or event-based triggers, to reduce unnecessary cloud transmissions while maintaining detection accuracy. Additionally, since experiment 4 and experiment 5 revealed that cloud inference remains a major bottleneck, further research could evaluate edge AI accelerators, such as the NVIDIA Jetson Nano or Google Coral TPU, to determine if they can offload more computationally intensive tasks from the cloud while still keeping latency and bandwidth usage manageable.

Another key area for further exploration is energy efficiency, which was not directly addressed in this study. Given the high latency observed in cloud-based processing (7+ seconds in experiments 3, 4, and 5), future research could assess the power consumption of different architectures, comparing the energy costs of running inference on high-performance cloud instances versus optimized edge models. This would provide deeper insight into the sustainability of edge computing in industrial settings. Additionally, while this thesis focused exclusively on AWS services, further work could evaluate multi-cloud solutions, where workloads are distributed between AWS, Google Cloud, and Azure to optimize cost, performance, and reliability. Experiment 4 showed that AWS Rekognition's 120-second limitation on video segments makes continuous real-time analysis difficult, so alternative cloud platforms might offer more flexible or cost-efficient solutions.

Finally, a more holistic cost analysis could be conducted by factoring in long-term operational expenses, such as cloud storage costs, data transfer fees, and edge device maintenance. While this study measured AWS service costs, an industry-wide economic feasibility study could provide a more concrete comparison between cloud-dependent architectures and edge-heavy deployments. By addressing these areas, future research could refine the balance between edge and cloud computing, making IIoT video stream processing more efficient, cost-effective, and scalable for real-world industrial applications.

References

[1] K. Khan, "A video streaming in industrial internet of things taxonomy (vsi-iot)", *International Journal of Multidisciplinary Research and Publications*, pp. 148–164, 2023.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges", *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] M. Satyanarayanan, "The emergence of edge computing", *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[4] L. S. Vailshery, *Edge computing market size worldwide 2026*, Jun. 2024. [Online]. Available: `https://www.statista.com/statistics/1175706/worldwide-edge-computing-market-revenue/`.

[5] J. Manyika, M. Chui, P. Bisson, *et al.*, "The internet of things: Mapping the value beyond the hype", 2015.

[6] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering", *Empirical Software Engineering*, vol. 20, pp. 1427–1455, 2015.

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications", *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[8] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges", *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[9] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions", *IEEE transactions on industrial informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[10] P. P. Gaikwad, J. P. Gabhane, and S. S. Golait, "A survey based on smart homes system using internet-of-things", in *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, IEEE, 2015, pp. 0330–0335.

[11] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey", *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[12] P. K. Malik, R. Sharma, R. Singh, *et al.*, "Industrial internet of things and its applications in industry 4.0: State of the art", *Computer Communications*, vol. 166, pp. 125–139, 2021.

[13] M. A. Rahman, M. S. Hossain, A. J. Showail, N. A. Alrajeh, and A. Ghoneim, "Ai-enabled iiot for live smart city event monitoring", *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 2872–2880, 2021.

[14] P. Zheng, X. Xu, and L. Wang, *Editorial notes: Industrial internet-of-things (iiot)-enabled digital servitisation*, 2023.

[15] I. Ungurean and N. C. Gaitan, "A software architecture for the industrial internet of things—a conceptual model", *Sensors*, vol. 20, no. 19, p. 5603, 2020.

[16] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "A comprehensive survey on interoperability for iiot: Taxonomy, standards, and future directions", *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–35, 2021.

[17] AWS, *What is edge computing?*, Accessed on December 9, 2024, 2024. [Online]. Available: `https://aws.amazon.com/what-is/edge-computing/`.

[18] P. Silva, A. Costan, and G. Antoniu, "Investigating edge vs. cloud computing trade-offs for stream processing", in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 469–474.

[19] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things", in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

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

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

[22] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance", in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 68–81.

[23] S. Singh, "Optimize cloud computations using edge computing", in *2017 International Conference on Big Data, IoT and Data Science (BID)*, IEEE, 2017, pp. 49–53.

[24] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey", *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[25] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey", *IEEE Access*, vol. 8, pp. 58 322–58 336, 2020.

[26] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges", *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[27] C. Ge, N. Wang, S. Skillman, G. Foster, and Y. Cao, "Qoe-driven dash video caching and adaptation at 5g mobile edge", in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 237–242.

[28] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming", *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1076–1089, 2017.

[29] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges", *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.

[30] N. Ministerrådet, *Nordic Public Sector Cloud Computing-a Discussion Paper*. Nordic Council of Ministers, 2012.

[31] B. K. Rani, B. P. Rani, and A. V. Babu, "Cloud computing and inter-clouds–types, topologies and research issues", *Procedia Computer Science*, vol. 50, pp. 24–29, 2015.

[32] G. Pallis, "Cloud computing: The new frontier of internet computing", *IEEE internet computing*, vol. 14, no. 5, pp. 70–73, 2010.

[33] M. Armbrust, A. Fox, R. Griffith, *et al.*, "A view of cloud computing", *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[34] P. Gupta, A. Seetharaman, and J. R. Raj, "The usage and adoption of cloud computing by small and medium businesses", *International journal of information management*, vol. 33, no. 5, pp. 861–874, 2013.

[35] AWS, *Amazon compute service level agreement*, Accessed on November 11, 2024, 2022. [Online]. Available: `https://aws.amazon.com/compute/sla/`.

[36] T. Lynn, G. Fox, A. Gourinovitch, and P. Rosati, "Understanding the determinants and future challenges of cloud computing adoption for high performance computing", *Future Internet*, vol. 12, no. 8, p. 135, 2020.

[37] R. Islam, V. Patamsetti, A. Gadhi, *et al.*, "The future of cloud computing: Benefits and challenges", *International Journal of Communications, Network and System Sciences*, vol. 16, no. 4, pp. 53–65, 2023.

[38] AWS, *Cloud computing with aws*, Accessed on November 11, 2024, 2024. [Online]. Available: `https://aws.amazon.com/what-is-aws`.

[39] AWS, *What is amazon ec2?*, Accessed on November 12, 2024, 2024. [Online]. Available: `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html`.

[40] AWS, *What is amazon s3?*, Accessed on December 9, 2024, 2024. [Online]. Available: `https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html`.

[41] AWS, *What is aws iot?*, Accessed on November 11, 2024, 2024. [Online]. Available: `https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html`.

[42] AWS, *What is aws iot greengrass?*, Accessed on November 11, 2024, 2024. [Online]. Available: https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html.

[43] AWS, *What is amazon kinesis video streams?*, Accessed on November 12, 2024, 2024. [Online]. Available: https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/what-is-kinesis-video.html.

[44] AWS, *What is amazon sns?*, Accessed on January 9, 2025, 2025. [Online]. Available: https://docs.aws.amazon.com/sns/latest/dg/welcome.html.

[45] AWS, *What is amazon simple queue service?*, Accessed on January 9, 2025, 2025. [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html.

[46] AWS, *What is aws lambda?*, Accessed on January 9, 2025, 2025. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[47] ISO, *Machine learning (ml): All there is to know*, Accessed on November 20, 2024, 2024. [Online]. Available: https://www.iso.org/artificial-intelligence/machine-learning.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[50] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[51] G. AI, *Litert overview*, Accessed on January 2, 2025, 2024. [Online]. Available: https://ai.google.dev/edge/litert.

[52] Keras, *About keras 3*, Accessed on January 2, 2025, 2025. [Online]. Available: https://keras.io/about/.

[53] Keras, *Keras applications*, Accessed on January 2, 2025, 2025. [Online]. Available: https://keras.io/api/applications/.

[54] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[55] OpenCV, *About*, Accessed on December 11, 2024, 2024. [Online]. Available: https://opencv.org/about/.

[56] J. Redmon, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[57] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context", in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[58] AWS, *What is amazon sagemaker ai?*, Accessed on December 9, 2024, 2024. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html.

[59] AWS, *What is amazon rekognition?*, Accessed on December 9, 2024, 2024. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html.

[60] AWS, *Amazon lookout for vision*, Accessed on January 8, 2025, 2025. [Online]. Available: https://aws.amazon.com/lookout-for-vision/.

[61] AWS, *Aws panorama*, Accessed on January 8, 2025, 2025. [Online]. Available: https://aws.amazon.com/panorama/.

[62] AWS, *Aws panorama devices*, Accessed on January 8, 2025, 2025. [Online]. Available: `https://aws.amazon.com/panorama/appliance/`.

[63] AWS, *Model deployment at the edge with sagemaker edge manager*, Accessed on January 28, 2025, 2025. [Online]. Available: `https://docs.aws.amazon.com/sagemaker/latest/dg/edge.html`.

[64] M. Ali, "Edge enhanced scalable and real-time video stream analytics", Ph.D. dissertation, University of Leicester, 2024.

[65] C. Savaglio, P. Mazzei, and G. Fortino, "Edge intelligence for industrial iot: Opportunities and limitations", *Procedia Computer Science*, vol. 232, pp. 397–405, 2024.

[66] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform", in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–13.

[67] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, "Video stream analysis in clouds: An object detection and classification framework for high performance video analytics", *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1152–1167, 2016.

[68] M. Ali, A. Anjum, O. Rana, A. R. Zamani, D. Balouek-Thomert, and M. Parashar, "Res: Real-time video stream analytics using edge enhanced clouds", *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 792–804, 2020.

[69] M. Ali, A. Anjum, M. U. Yaseen, *et al.*, "Edge enhanced deep learning system for large-scale video stream analytics", in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, IEEE, 2018, pp. 1–10.

[70] Raspberry-Pi-Foundation, *Raspberry pi*, Accessed on November 21, 2024, 2024. [Online]. Available: `https://www.raspberrypi.com`.

[71] AWS, *Understanding rekognition's types of analysis*, Accessed on January 21, 2025, 2025. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/how-it-works-types.html.

[72] AWS, *Amazon kinesis video streams cpp producer, gstreamer plugin and jni*, Accessed on December 19, 2024, 2024. [Online]. Available: https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.

[73] gstreamer, *What is gstreamer?*, Accessed on December 19, 2024, 2024. [Online]. Available: https://gstreamer.freedesktop.org.

[74] R. S. Barnes, *Udp vs tcp, how much faster is it? [closed]*, Accessed on February 11, 2025, 2009. [Online]. Available: https://stackoverflow.com/questions/47903/udp-vs-tcp-how-much-faster-is-it.

[75] AWS, *Amazon kinesis video streams pricing*, Accessed on January 22, 2025, 2025. [Online]. Available: https://aws.amazon.com/kinesis/video-streams/pricing/.

[76] AWS, *Amazon rekognition pricing*, Accessed on January 22, 2025, 2025. [Online]. Available: https://aws.amazon.com/rekognition/pricing/.

[77] AWS, *Amazon sns pricing*, Accessed on January 22, 2025, 2025. [Online]. Available: https://aws.amazon.com/sns/pricing/.

[78] AWS, *Amazon sqs pricing*, Accessed on January 22, 2025, 2025. [Online]. Available: https://aws.amazon.com/sqs/pricing/.

[79] AWS, *Amazon sagemaker ai pricing*, Accessed on January 29, 2025, 2025. [Online]. Available: https://aws.amazon.com/sagemaker-ai/pricing/.

Appendix A

Usage of Generative AI for Language Improvement

In the making of this thesis, generative AI (ChatGPT 4o) has been used for language improvement. The following steps will describe its use:

1. Write paragraph.
2. Ask ChatGPT, could this paragraph be written using better English language.
3. Check that the content is still as you want it to be and that the point you want to make is not distorted.
4. Use the improved version of the paragraph, if the content is not distorted.

This tactic has several benefits. First of all, it helps the writer to learn how to write better English. Second, usually the text becomes more readable, and the overall flow of the text becomes clearer. Of course, this will depend on the writer but at least on my case, usually the improved version was better and that's why I ended up using the improved version in several cases.