jamk

Enhancing IoRT with Multimodal AI: A Framework for LLMs, Computer Vision, and Predictive ML

Md Ashikur Rahman Shourov

Master’s Thesis
December 2025
Master's Degree Programme in Robotics

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Shourov, Md Ashikur Rahman

Enhancing IoRT with Multimodal AI: A Framework for LLMs, Computer Vision, and Predictive ML

Jyväskylä: Jamk University of Applied Sciences, December 2025, 57 pages.

Degree Programme in Robotics. Master's thesis.

Permission for open access publication: No

Language of publication: English

Abstract

Artificial intelligence is the latest trend to appear in industrial and connected systems, but most of its available solutions are based on a single data type, such as text, pictures, or sensor readings. This restricts their ability to understand complex situations. This thesis proposed a lightweight multimodal and aimed at designing and testing a multimodal framework that incorporates large language models, computer vision and predictive machine learning within an Internet of Robotic Things (IoRT) system. The idea was to investigate the possibility of these three components of AI to work together on edge hardware while maintaining low-latency and acceptable accuracy.

The research-based approach was followed in the development work. Lightweight versions of DistilBERT, YOLOv8n and a Random Forest classifier were chosen and combined together using an event-driven pipeline. The implementation of the system was done using Raspberry Pi 5 and the modules communicated via MQTT. The system was tested with the help of public datasets, open-source tools as well as with the help of synthetic logs of IoRT sensors. The metrics used to measure performance were accuracy, latency, resource usage, and stability of the entire system when operating under real-time conditions.

The result showed that the system handled commands, detected objects, and found abnormal sensor values all at once. The combined pipeline also reacted in a short time window and remained stable even in the presence of noise in the inputs. The multimodal structure was found to be superior to single modes, enhancing the quality of decisions and decreasing wrong actions.

The study concludes that edge devices can implement multimodal intelligence with the assistance of lightweight AI models. It has the possibility to expand this framework with more sensors, enhanced fusion logic or more powerful edge accelerators. It is also possible to repeat the development process of the work by students and engineers who are interested in creating similar IoRT prototypes.

Keywords/tags (subjects)

Internet of Robotic Things (IoRT), Multimodal AI, Large Language Models (LLMs), Computer Vision, Machine Learning, Edge Computing.

Miscellaneous (Confidential information)

No confidential information.

Contents

Figures

Tables

1 Introduction

The use of AI tools is becoming common in various fields of work and research. They are used to assist machines in carrying out what only humans did, like interpreting commands, viewing camera information and processing sensor readings. In most areas, machines can now automatically make decisions and perform computations in response to a variety of tasks, thanks to recent advances in the field of data-driven techniques, networking, and robotics. With the pace of technological progress, there is a necessity to find ways of integrating various methods of AI and implementing them in the real world. This introduction provides the background, scope of the study, objectives and problem statement of work and outlines its context.

1.1 Background and Motivation

The field of artificial intelligence is defining the new technologies and affecting most spheres of modern society, offering solutions that involve perception and reasoning, as well as decision-making. Enhanced large language models, vision models, and predictive machine learning can now be used to create intelligent systems that can chat with users, comprehend camera scenes, and predict the future (LeCun, Bengio, & Hinton, 2015; Goodfellow, Bengio, & Courville, 2016). Although such breakthroughs are made, many systems today utilize either one type of data at a time. As an illustration, one system can just be text-based, another based on imagery, and a third based on sensors. As an example, BERT and GPT models have advanced natural language processing to record high accuracy (Devlin et al., 2018; Brown et al., 2020), whereas models like convolutional neural networks and YOLO have high accuracy in image recognition (Krizhevsky, Sutskever, & Hinton, 2012; Redmon et al., 2016). Machine learning has also been extensively utilized in predictive analytics, especially in maintenance, healthcare, and logistics (Chen & Ran, 2019). Most existing systems can only utilize one model at a time and thus are not quite flexible when the environment changes.

Table 1 presents the key differences between single and multimodal IoRT solutions.

Approach	Typical Application	Strengths	Limitations
NLP / LLM only	Voice assistants, command parsing	Natural interaction with users	No visual or predictive awareness
Computer Vision only	Object detection in robotics, quality inspection	Accurate perception of environment	Cannot interpret language or predict events
Predictive ML only	Predictive maintenance, anomaly detection	Anticipates failures and trends	Lacks perception and natural communication
Multimodal AI (NLP + CV + ML)	Integrated IoRT systems	Combines perception, prediction, and interaction	Resource intensive and requires optimization

Table 1. Comparison of single-modal and multimodal approaches in IoRT.

The Internet of Robotic Things (IoRT) is a more expanded version of the Internet of Things (IoT) that incorporates autonomous robots and smart devices in networks that have the ability to sense, analyze and act together. IoRT also ensures progress in industrial automation, agriculture, medical care, and logistics (García et al., 2020; Frontiers, 2020). However, the vast majority of IoRT applications still use narrow AI, in which only a single modality (language, vision, sensor data, etc.) is handled at a time. This limits the effectiveness and responsiveness of the system, especially in real-time situations, where multimodal interpretation is required.

This study is driven by the increasing curiosity regarding the need to have multimodal AI systems that can run on edge devices, integrating language, vision, and predictive analytics under resource limited environments. By combining these modalities, the IoRT systems would be able to see, listen, and anticipate at the same time and consequently make prompt and more dependable decisions. This functionality is especially needed in sectors that are resource-demanding and safety-critical like manufacturing, autonomous robotics, and intelligent healthcare (Zhou, Chen, & Li, 2020).

The recent advances in edge computing and the development of lightweight AI models have demonstrated that high-performance systems do not necessarily need to have heavy and cloud-based resources. Reduced models, like MobileNet in vision tasks and distilled language models in NLP, have the potential to be compatible with efficient performance and still have an acceptable level of accuracy (Howard et al., 2017; Sanh et al., 2019). This opens a possibility of designing an IoRT framework that combines the use of LLMs, computer vision, and predictive ML in real-time, without excessive resource constraints. The work helps fill this gap and leads to a better understanding of the fact that the multimodal AI integration at the edge is not only possible but also effective.

The multimodal system utilized in the current work is a combination of language, vision, and sensor data. The high-level view of this combination is described in Figure 1 which illustrates how the three modules support one another.

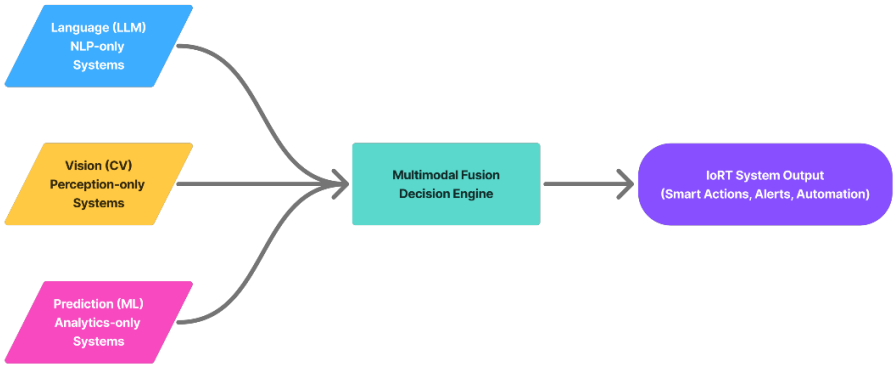


Figure 1. Multimodal AI integration in IoRT.

1.2 Problem Statement

Internet of Robotic Things (IoRT) has been popularly researched to be used in automation, healthcare, and smart infrastructure, yet most systems are based on single-modal intelligence. Current systems usually combine either computer vision, predictive machine learning, or natural language processing separately (García et al., 2020; Chen and Ran, 2019). Although these methods yield satisfactory performance in specific tasks, they are not flexible in dynamic real-life situations where multimodal input is supposed to be handled simultaneously.

The primary challenge is that the existing IoRT architecture is unable to unite language understanding, perception-based vision, and predictive analytics in either near-real-time or low-latency systems and on resource-constrained platforms. Models like GPT-4 size LLMs or models that train with high capacity vision models like YOLOv8 require computation resources that are beyond the typical edge device capabilities (Zhou, Chen, and Li, 2020).

Table 2 provides the comparisons between the various AI models and edge devices in relation to memory requirements, computing power, and edge deployability.

Model Type	Example Model	Typical Resource Needs	Suitability for Edge Devices
Large Language Models (LLMs)	GPT-3 (175B parameters), GPT-4	> 300 GB memory, multi-GPU clusters, high power consumption (Brown et al., 2020)	❌ Not suitable without cloud offloading
Lightweight LLMs	DistilBERT, TinyBERT	~200–400 MB memory, can run on CPU with optimization (Sanh et al., 2019)	✔ Possible with quantization/pruning
Computer Vision (CV)	YOLOv8, Faster R-CNN	High-end GPU required, latency ~30–60 ms per frame at inference (Redmon et al., 2016)	⚠ Limited without accelerators
Lightweight CV	MobileNet-SSD, YOLOv8n	<100 MB memory, can achieve ~20–30 fps on Jetson/Coral devices (Howard et al., 2017)	✔ Suitable for real-time edge inference
Predictive ML	Random Forests, SVM, small NN	Low to moderate memory (~10–50 MB), CPU-friendly, fast inference	✔ Highly suitable
Edge Hardware	Raspberry Pi 5, NVIDIA Jetson Nano, Coral TPU	RAM 2–8 GB, limited GPU/TPU cores, power-efficient (Zhou et al., 2020)	Requires lightweight/optimized models

Table 2. Resource requirements of AI models compared with edge hardware capabilities.

In consequence, the majority of systems either push processing to the cloud, add latency and reliability threats or limit their functionality to a single modality at a time.

Due to this, IoRT devices are not able to take action autonomously and comprehend the entire situation surrounding them. In the absence of multimodal integration, robots will not be able to interpret the instructions, sense the environment and predict changes in real time. To address this, there is a need of a framework that will allow effective integration of lightweight LLMs, computer vision models, and predictive ML on the edge with an acceptable level of accuracy, low latency and acceptable power usage to support near real-time application.

1.3 Research Objectives and Questions

The concept of combining multimodal AI with the Internet of Robotic Things (IoRT) systems has been identified as the research gap that is truly picking up. It is rapidly growing in popularity, particularly in cases when edge devices require processing various data streams effectively (Sharshar, Khan, Ullah, & Guizani, 2025; Zhou, Chen, & Li, 2020). Although previous literature has shown that individual modalities like language, vision, or prediction have improved, there still are few comprehensive frameworks integrating all the three and implemented in real-time tasks (Chen &

Ran, 2019). To fill this gap, the paper was focused on designing and testing a lightweight and modular IoRT architecture with the ability to combine large language models, computer vision, and predictive machine learning through an event-driven pipeline.

The following objectives guided the research:

- To explore the present multimodal artificial intelligence situation and the appropriate lightweight models that can be used in natural language understanding, visual perception, and predictive analytics.
- To create a modular IoRT design that enables the event-based communication between the LLMs, computer vision systems, and machine learning models.
- To install and test the suggested architecture with the help of simulated IoRT data and edge-capable devices or with virtual machines.
- To compare the system with single-modal methods in terms of resource usage, latency and accuracy of the system.
- To examine the advantages and obstacles of integration of multimodals within the IoRT and suggest suggestions regarding future progression.

According to these objectives, the following research questions were developed:

1. What is the way to implement lightweight LLMs, computer vision and predictive ML models to an IoRT system running on resource-constrained edge hardware?
2. Which are the efficient lightweight models that can be used to do real-time IoRT applications on edge devices?
3. Does multimodal integration make better decisions than single-modal systems?
4. What are the key weaknesses of multimodal AI on edge devices and what are their impacts on the system performance?

These questions and aims offered a systematic direction to the research, both in analytical terms and in the implementation stages.

1.4 Scope and Limitations

The work focused on the design, implementation, and evaluation of a lightweight multimodal framework of the Internet of Robotic Things (IoRT). The research was based on the synthesis of

three mutually supportive parts of artificial intelligence, including large language models (LLMs) to understand natural language, computer vision to recognize objects, and machine learning to predict analytics. The focus of the work was to prove the functionality in real-time of a simulated or edge-capable environment, and not on a full-scale industrial application.

The areas of implementation involved:

1. Choosing of light weight AI models (e.g., DistilBERT, YOLOv8n, Random Forests) that could be deployed on edge computing.
2. Creation of a pipeline that is modular and event driven so that it can communicate between multimodal components.
3. Training and testing of models with datasets that are publicly available and simulated log files of the IoRT sensors.
4. Evaluation of accuracy, latency, and resources use as vital performance measurements.

The limitations of the work were identified as follows:

- **Data limitations:** Open-source and synthetic sensor logs were the only data that were employed; no proprietary or industrial data were used.
- **Hardware scope:** Testing and implementation were performed on Raspberry Pi 5 Model B as the primary edge-AI device. No dedicated GPU or TPU accelerators were used. All inference tasks were executed by CPU-based execution. It can be further developed in the future by using Jetson Nano or Coral TPU to enhance vision performance.
- **Model scope:** Training large-scale LLMs or CV models from scratch was excluded due to computational constraints. Only pre-trained and fine-tuned models were considered.
- **Contextual scope:** The work focused on general IoRT applications rather than domain-specific applications such as autonomous vehicles, precision and surgical robotics.
- **Cloud integration:** While optional cloud assistance was considered, continuous dependence on cloud services was excluded to maintain focus on edge deployment feasibility.

These limits minimized the scope of the study to be conducted within the time and resource constraints, without losing the results that will contribute towards the understanding of multimodal AI integration in IoRT. Thus, the analysis is aimed at feasibility characteristics instead of the highest accuracy standards or industrial grade durability.

In order to clarify research boundaries, Table 3 is a summary of what is within the study and what is outside the study.

Area	Scope (Included)	Limitations (Excluded)
Data	Open-source datasets (text, images) and simulated IoRT sensor logs	Proprietary or industrial datasets
Hardware	Raspberry Pi 5 Model B edge-device	Other edge AI accelerators (Jetson Nano, Coral TPU, TPU hats, NPU shields) not tested but considered for better performance
Models	Pre-trained and lightweight AI models (DistilBERT, YOLOv8n, Random Forests)	Training large-scale LLMs or CV models from scratch
Context	Generic IoRT scenarios for proof-of-concept	Domain-specific optimization (autonomous vehicles, agriculture, medical robotics, etc.)
Cloud	Optional cloud-assisted services for non-critical tasks	Continuous dependence on cloud processing; cloud-centric deployment

Table 3. Scope and limitations of the research.

1.5 Structure of the Thesis

The work is divided into 7 major chapters, references, and appendices. The first chapter is an introduction to the background, motivation, research problem, objectives, scope and limitations. The second chapter outlines the theoretical framework and the literature review, including the large language models, computer vision, machine learning, and the Internet of Robotic Things. Chapter three explains the research methodology, such as the research design, data sources, tools, and evaluation metrics.

Chapter four introduces system development and implementation with the description of the architecture, multimodal elements integration, and issues faced. Chapter five presents the experimental findings and analysis and compares them to the baseline methods and discusses the findings. In chapter six, the findings are discussed by connecting them with earlier studies and research questions. Finally, Chapter seven brings to an end the thesis and gives suggestions concerning future work.

The last section of the thesis includes the reference list that is made in the APA 7 format and appendices, where supporting material in the form of sample data, code fragments, and other figures can be found.

2 Theoretical Framework and Literature Review

The chapter outlines the theoretical background and literature review that lays the groundwork of the study. The purpose is to explain the most important ideas and previous studies concerning the Internet of Robotic Things (IoRT) and its connection with multimodal artificial intelligence. The review discusses the use of large language models to perform natural language processing, computers vision to perform perception, machine learning to perform predictive analytics, and the use of edge computing to enable low-latency performance. In every subsection, the modern level of knowledge is noted, the technological prospects are identified, and the limitations serving as the driving force of the study are outlined. Combining these views, the chapter creates the academic and practical background of the construction of a multimodal IoRT framework.

2.1 Overview of Internet of Robotic Things (IoRT)

IoRT is IoT-based, with the addition of robots capable of identifying information in the environment and acting on it. IoRT unites robots, sensors, and communication devices to enable them to operate in teams and respond to changes without direct human intervention (García, Alonso, Prieto, and Corchado, 2020). Compared to conventional IoT systems where data collection and transmission are primarily achieved by means of monitoring, IoRT incorporates robotics to actively adapt to environmental changes, which is makes it suitable in the dynamic domains. Table 4 provides sample IoRT application in other fields and indicates their key advantages and difficulties.

Domain	Example Use Case	Benefits	Challenges
Manufacturing	Smart factories with robotic arms connected to IoT sensors	Improved efficiency, predictive maintenance, reduced downtime	High integration costs, cybersecurity risks
Healthcare	Assistive robots and remote patient monitoring	Real-time health data, improved patient care	Data privacy, safety certification requirements
Agriculture	Autonomous drones and tractors with IoT-enabled sensors	Precision farming, optimized resource use	Connectivity in rural areas, hardware durability
Logistics	Warehouse robots and automated delivery systems	Faster operations, reduced labor costs	Scalability, network reliability
Smart Cities	IoRT-enabled surveillance and traffic management	Enhanced public safety, better traffic flow	Ethical concerns, data management complexity

Table 4. Applications of IoRT across domains.

The architecture of IoRT typically includes three layers:

1. **Perception layer:** sensors, cameras, and robotic units that collect raw data.
2. **Network layer:** communication technologies such as 5G, Wi-Fi, and MQTT protocols for real-time data exchange.
3. **Application layer:** AI algorithms and decision-making components that process data and trigger actions (Frontiers, 2020).

There are multiple layers in which devices operate in IoRT. These layers reveal the connection between sensing, communication and applications. These layers are shown in simple form in figure 2.

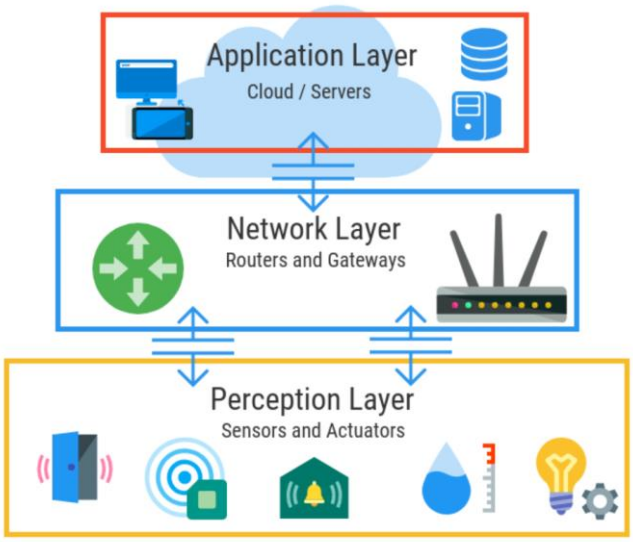


Figure 2. IoRT layered architecture consisting of perception, network, and application layers.

IoRT has a few shortcomings despite its potential, such as computational resources and power consumption, and latency with advanced AI models. Existing IoRT implementations tend to use cloud-based computing, creating delays and dependence on network availability (Zhou, Chen, & Li, 2020). This puts the emphasis on edge-capable solutions in which one can process AI workload locally without compromising performance.

2.2 Large Language Models (LLMs) in Embedded Systems

Text processing models such as BERT and GPT have transformed the way computers process text since they are capable of processing short messages and producing transparent answers (Devlin, Chang, Lee, & Toutanova, 2018; Brown et al., 2020). They are also good at intent recognition, dialogue management, summarization, and contextual reasoning, which are applicable to IoRT systems that require spoken or textual commands to be processed in real time.

Full-scale LLMs are however resource intensive. As an example, GPT-3 has 175 billion parameters, and it needs high-performance GPUs and hundreds of gigabytes of memory (Brown et al., 2020). The fact that these models require powerful hardware means that they often cannot be run on small computers unless some processing is offloaded to the cloud which can add to latency and decrease system reliability (Zhou, Chen, & Li, 2020).

In response, the researchers have come up with lightweight versions of LLM models that are suitable to a limited set of resources. DistilBERT is 40 percent smaller than BERT and still maintains most of its accuracy (Sanh, Debut, Chaumond, & Wolf, 2019). Equally, both TinyBERT and MobileBERT are even smaller models that are applicable in real-time inference on CPUs and edge accelerators (Jiao et al., 2020; Sun et al., 2020). Such models allow it to be possible to incorporate natural language processing into the IoRT systems without straining the available resources.

Table 5 compares big and small language models and justifies which models are more realistic to use with edge devices with IoRT.

Model	Parameters (approx.)	Size (MB)	Inference Speed	Edge Suitability
GPT-3	175B	> 350,000	Slow (requires multi-GPU cluster)	❌ Not suitable
BERT (base)	110M	~400	Moderate (GPU required)	⚠️ Limited
DistilBERT	66M	~200	Fast (CPU-friendly with quantization)	✔ Suitable
TinyBERT	66M (compressed)	~120	Fast (mobile CPUs/TPUs)	✔ Suitable
MobileBERT	25M	~100	Fast, optimized for mobile devices	✔ Suitable

Table 5. Comparison of large-scale and lightweight LLMs for edge deployment in IoRT.

Model sizes and memory requirements should be compared to choose an appropriate language model to use on the edge devices. This comparison is visualized in figure 3.

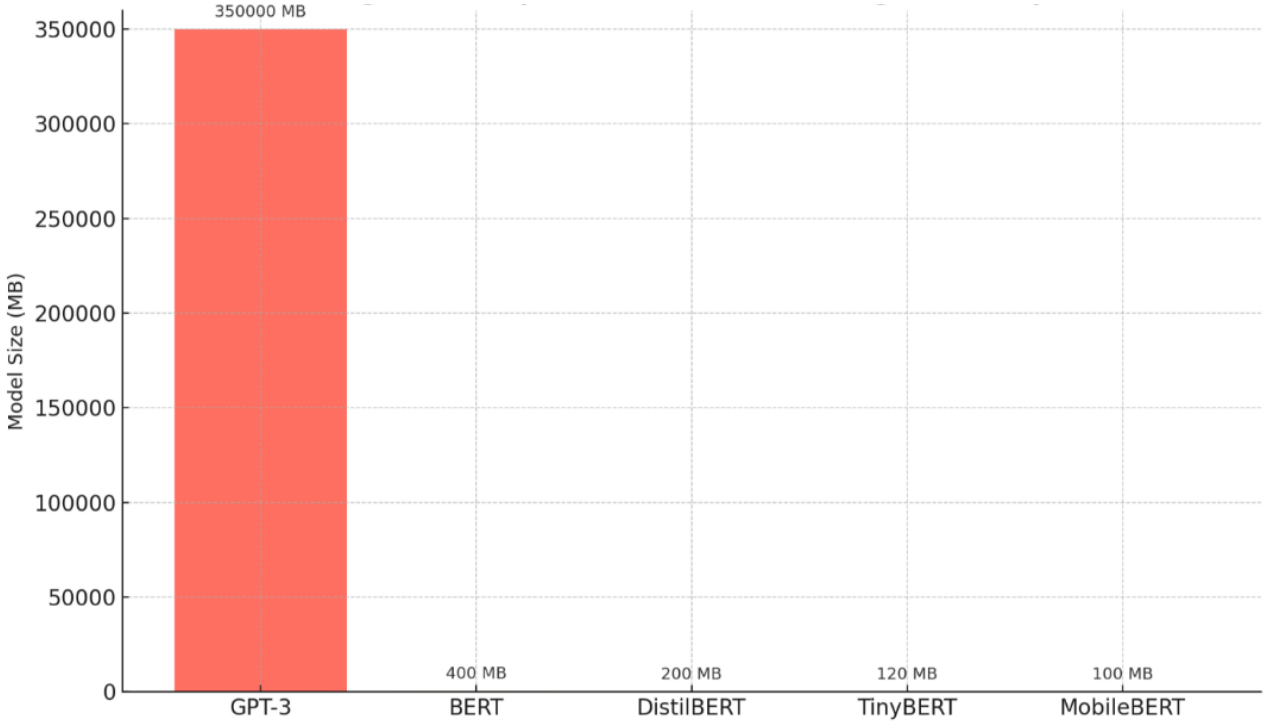


Figure 3. Comparison of LLM sizes highlighting suitability for edge deployment.

The application of LLM in embedded IoRT systems is often an intent classifier and a command parsing system as opposed to a full-scale dialogue generator. Such limited usage will make the models efficient yet still allow them to interact with robots in a natural way. New developments in the fields of quantization, pruning, and hardware acceleration will bring more advantages to implementing LLMs at the edge (Han, Mao, & Dally, 2016). In constrained systems, LLM modules are usually limited to intent classification, command interpretation, and template-bounded generation to maintain predictable latency.

2.3 Computer Vision for Perception and Context Understanding

Computer vision (CV) plays a significant role in the field of IoRT since it allows machines to perceive the environment, locate objects and comprehend the situation. The CNN-based models such as AlexNet initiated a significant advance in image recognition, and subsequent models extended this advance (Krizhevsky, Sutskever, & Hinton, 2012), which proved the strength of the deep

architecture on massive datasets. Later models, including ResNet (He, Zhang, Ren, & Sun, 2016) and Inception (Szegedy et al., 2015) were more accurate and scaled.

In real-time IoRT applications, such object detection models as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) are also popular because of their speed and accuracy (Redmon et al., 2016; Liu et al., 2016). Nevertheless, more advanced models like YOLOv8 or Faster R-CNN require the use of expensive GPUs, and will not suit edge devices.

Embedded and mobile environments have been developed to have lightweight alternatives. Models like MobileNet-SSD (Howard et al., 2017), YOLO-Nano (Wang et al., 2020), and SqueezeNet (Iandola et al., 2016) are much smaller in terms of their parameters and have a high level of accuracy. With these optimizations, the IoRT systems can do real-time detection on devices such as NVIDIA Jetson Nano, Google Coral TPU or Raspberry Pi with accelerators.

Table 6 lists the strengths and weaknesses of various computer vision models.

Model	Parameters (approx.)	Size (MB)	Speed (FPS)	Edge Suitability
AlexNet	60M	~240 MB	~200 (GPU)	⚠ Limited - outdated, large
ResNet-50	25M	~100 MB	~90 (GPU)	⚠ Moderate - accurate but heavy
YOLOv3	62M	~240 MB	~45 (GPU)	❌ Too heavy for most edge
YOLOv4	64M	~250 MB	~60 (GPU)	❌ Requires high-end GPU
YOLO-Nano	4M	~12 MB	~25–30 (Jetson)	✔ Suitable
MobileNet-SSD	5.7M	~17 MB	~20–25 (Jetson/Coral)	✔ Suitable
SqueezeNet	1.2M	~5 MB	~25–30 (CPU)	✔ Suitable for ultra-low power

Table 6. Comparison of computer vision models for IoRT edge deployment.

When choosing a vision model, it is important to check three things: the speed at which the model will run, precision of the model and the size of the model. This trade-off is summarized in figure 4.

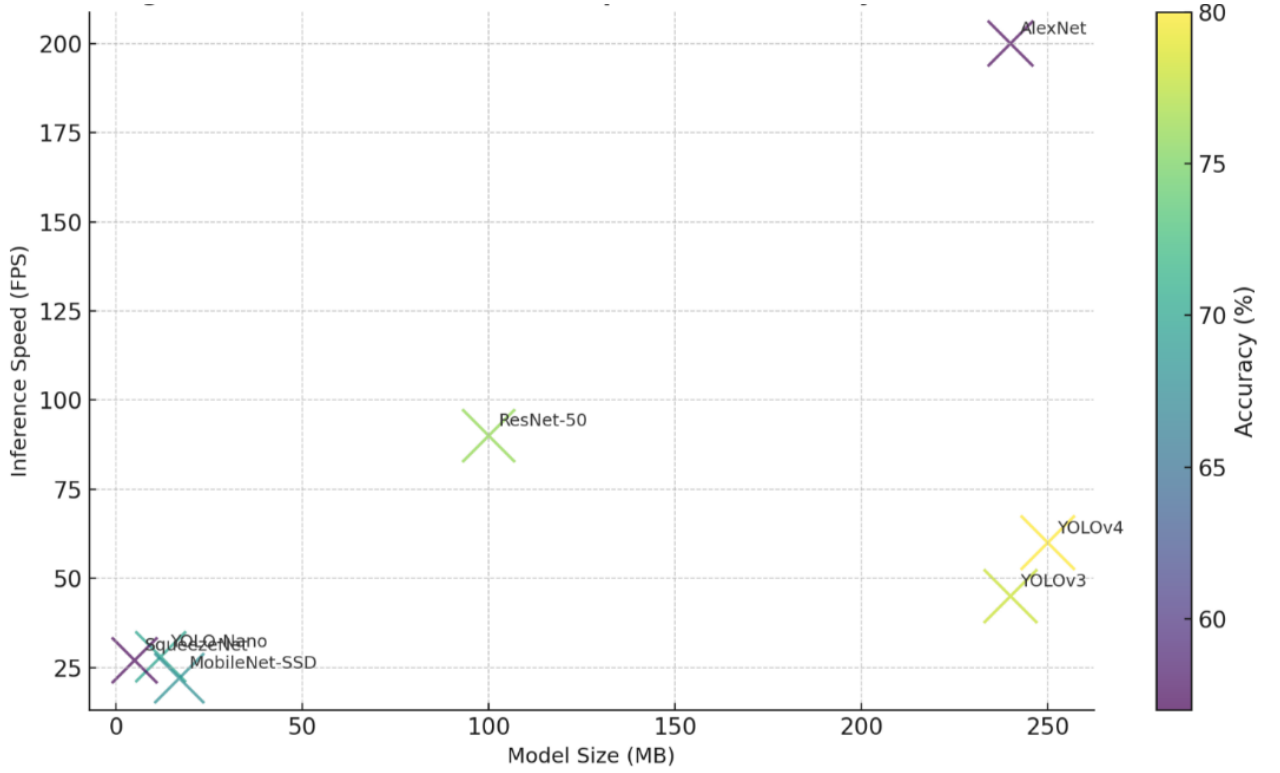


Figure 4. Trade-off between model size, speed, and accuracy in selected CV models.

Tasks performed with the help of CV in IoRT include obstacle avoidance, quality control in the manufacturing industry, patient monitoring in health care, and surveillance in intelligent cities (García et al., 2020). The primary issue is to be able to reach high accuracy without compromising latency and efficiency. Edge deployment involves using a wise choice of models, quantization, and even frame-skip techniques to sustain performance in real-time (Han, Mao, & Dally, 2016).

2.4 Machine Learning for Predictive Analytics

Predictive models enable the IoRT devices to guess the future and observe unusual sensor patterns and respond faster. IoRT devices can predict failures, optimize performance, and react in advance by examining time-series data gathered by sensors (Chen & Ran, 2019). It can be applied in predictive maintenance in the manufacturing industry, in patient monitoring in the healthcare industry, in energy demand forecasting in smart grids, and in detecting anomalies in logistics systems (Zhou, Chen, & Li, 2020).

Various machine learning tools have been popular in predictive tasks in the context of IoRT:

- **Random Forests:** The interpretable and robust ensemble-based models. Anomalies and predictive maintenance are easy to detect.
- **Support Vector Machines:** Ideal when one is handling small or medium-sized datasets to achieve classification tasks. However, it is not as scalable with data of a high dimension and in real time.
- **Artificial Neural Networks:** Capable of picking up complex nonlinear patterns, can be implemented in light weights, used in the time-series prediction.
- **Recurrent Neural Networks and Long Short-Term Memory Networks:** Good in sequential sensor input. They are highly accurate, but they are difficult to optimize to run on the edge.
- **Lightweight Deep Models (1D-CNNs, TinyRNNs):** A new architecture targeting embedded devices. The architecture sacrifices a little bit of accuracy in order to be fast in making inferences.

Table 7 summarizes several predictive models and shows which ones are suitable for IoRT anomaly detection and forecasting tasks.

Model	Strengths	Limitations	Edge Suitability
Random Forests	Robust, interpretable, low training cost	Larger memory footprint with many trees	✔ Good
SVM	High accuracy on small/medium data	Poor scalability for very large datasets	⚠ Limited
ANN (shallow)	Learns nonlinear patterns, adaptable	Less accurate than deep models	✔ Good
RNN/LSTM	Handles sequential time-series well	Computationally heavy for edge devices	⚠ Limited
1D-CNN/TinyRNN	Lightweight, optimized for embedded forecasting	Slight drop in accuracy	✔ Suitable

Table 7. Comparison of machine learning models for predictive analytics in IoRT.

Figure 5 shows the general workflow of predictive analytics in the Internet of Robotic Things (IoRT), including all the stages between the gathering of sensor data and the identification of anomalies or the implementation of forecasting functions.

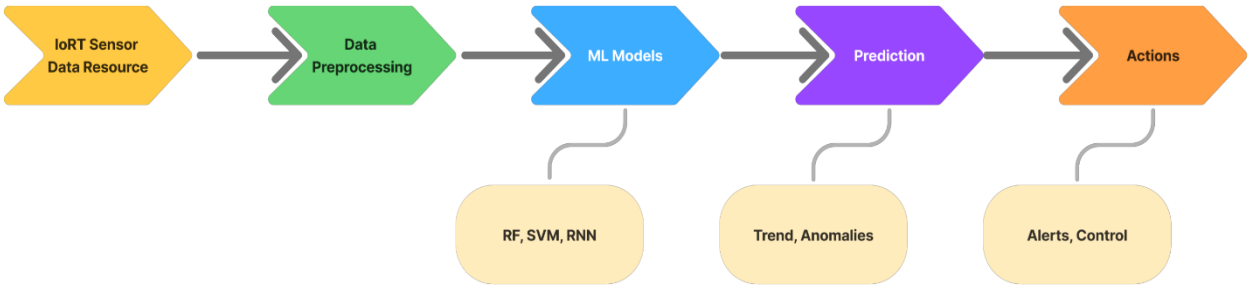


Figure 5. Workflow of predictive analytics in IoRT.

The difficulty is to find a balance between predictive accuracy and efficiency of computation. Classical ML algorithms like the Random Forests and SVMs can be easily processed using CPUs and can be applied to IoRT. Neural networks have greater accuracy but compression of models, quantization, or special accelerators are necessary to run on edge devices (Han, Mao, & Dally, 2016).

2.5 Edge AI and Real-Time Integration Challenges

The use of various forms of AI in IoRT requires the system to remain fast, as well as consume minimal memory and power. Cloud servers are beneficial in terms of their computing capabilities, but overutilizing them may result in a slower response and cause problems in case the network is not stable or secure (Zhou, Chen, & Li, 2020). In real-time IoRT systems, including industrial automation, autonomous robots, or healthcare monitoring, the latency has to be maintained in the milliseconds range, which is why cloud-only solutions are not feasible (Satyanarayanan, 2017).

Resource constraints are the first challenge. Complete models of AI, like GPT-3 or YOLOv4 models, require hundreds of megabytes through gigabytes of RAM, extremely high GPU and performance, which most edge devices cannot handle (Brown et al., 2020; Redmon et al., 2016). Even though the lightweight models, such as DistilBERT, YOLOv8n, or tiny RNNs, have been developed, still the challenges of executing them all together in a multimodal (language, vision, prediction) pipeline may still overwhelm the low-power hardware.

System integration is another barrier. LLMs, computer vision networks and predictive ML have a common design approach as individual, independent modules with varying data formats, latencies and optimization strategies. Integrating these into a unified event-driven pipeline needs tasks to be

carefully synchronized, inputs need to be synchronized, and effective communication protocols need to be utilized including MQTT or ROS 2 (Quigley et al., 2009).

Energy efficiency further limits adoption. There are numerous IoRT systems that use battery operated robots or mobile devices. Running several AI models simultaneously consumes more energy, decreasing the operational time and scalability. Methods like pruning, quantization, and hardware acceleration help alleviate this problem but usually at the price of a lower accuracy (Han, Mao, & Dally, 2016).

Lastly, reliability and security are also important issues. Multimodal systems should be able to operate resistantly to imperfect information environments like distorted sensor measurements, or incomplete input. Furthermore, edge AI introduces systems to security risks, such as adversarial attacks on models and data manipulation throughout communication (Dong et al., 2020).

Figure 6 sums up the primary technical and practical issues that emerge once multimodal AI is integrated into IoRT systems.

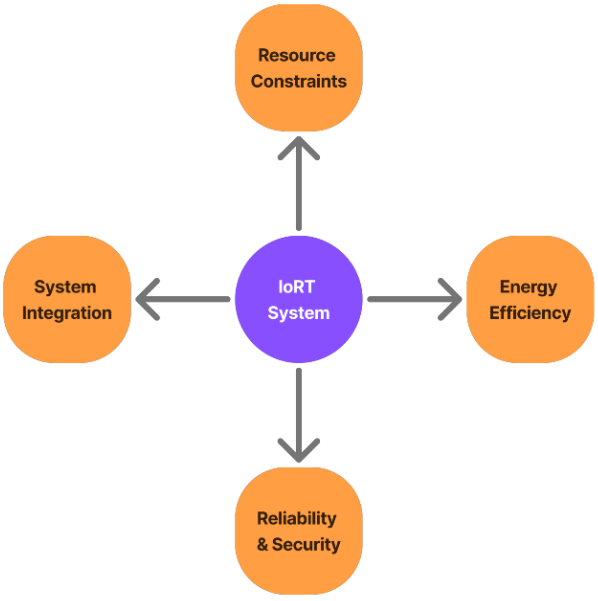


Figure 6. Key challenges of multimodal AI integration in IoRT.

These issues point to the fact that multimodal AI integration in the context of IoRT is not explored thoroughly in practice. They need to be addressed not just with optimized models but also with a

modular architecture which will be able to distribute tasks efficiently and with a near real-time performance. This gap inspires creating a lightweight framework integrating the use of LLMs, computer vision, and predictive ML as one IoRT system.

2.6 Summary and Research Gap

The literature review identified several technological breakthroughs that are applicable in the integration of artificial intelligence in the Internet of Robotic Things (IoRT). IoRT builds on the conventional IoT systems with the addition of robots and allows independent sensing, decision-making, and taking action across manufacturing, healthcare, agriculture, and logistics. Large language models (LLMs) have generally changed the nature of natural language processing, enabling machines to read and act on human commands. Real-time perception has become possible with the computer vision architecture, such as CNNs and YOLO-based models. Machine learning methods, such as Random Forests to recurrent neural networks, can be applied in predictive machine learning to aid in proactive decision making by detecting anomalies and trends. Lastly, edge computing has introduced the possibilities of bringing AI to the devices, minimizing latency and dependency on cloud services.

In spite of those developments, there are still serious obstacles in multimodal AI integration in the case of IoRT. Current systems usually implement LLMs, CV, or predictive ML individually, without making adaptations in dynamic systems. Full-scale models are very precise and they require resources beyond the capabilities of the edge devices. There are also lightweight variants, but the combination of the lightweight variant into one, event-driven pipeline is not well-explored. Other obstacles are cross-modularity synchronization, energy savings in the mobile devices, and resiliency in the presence of unreliable information or hostile environments.

The only gap in the existing literature is a simple, lightweight IoRT system with the possibility to operate with LLMs, vision models, and prediction models simultaneously in real time. To fill this gap it is necessary to choose effective model variants, optimize them around edge hardware and coordinate their interaction with a modular architecture. By addressing this issue, the work will seek to add a practical step toward the realization of multimodal IoRT systems that are resource-efficient and effective in the real world.

Even though numerous previous studies have investigated single elements of IoRT, including lightweight LLM or edge-based computer vision, few studies ever experimented with these tools combined on a real machine. Majority of the studies concentrate on cloud solutions or high level frameworks without regarding real latency, resource consumption or fusion behaviour on edge hardware. This leaves a loophole between theory and practical performance mostly in areas where quick decisions are needed. This thesis results would help fill this gap by analyzing a complete multimodal system directly on a Raspberry Pi 5. It is a good example of what can be achieved with inexpensive equipment and where further investigation is necessary.

3 Research Methodology

The study was conducted as a research-based development project which integrated theoretical study with practice. A methodology was developed to fill the identified research gap by developing and experimenting with a lightweight multimodal IoRT framework. The approach emphasized modularity, low-latency operation, and resource efficiency which ensure that large language models, computer vision systems, and predictive machine learning could operate together on edge-capable devices.

The chapter describes the research design and strategy, and later the system architecture and implementation environment. The data collection methods, model selection and evaluation are also provided. Moreover, the chapter describes the performance metrics that are to be applied to evaluate the system and covers the ethical issues associated with the work. This methodological ground accorded by the chapter shows how the research was designed in such a way that it would not only be scientifically rigorous, but also relevant to practice.

3.1 Research Design and Strategy

This work was developed using a research-based development approach. It is an appropriate option in engineering work, when we require not only new knowledge, but also a working system (Oates, 2006; Robson & McCartan, 2016). The design was a combination of exploration research, to examine the prospects of multimodal AI in IoRT, and experimental development, to design and conduct a prototype system. This two-fold attention made sure that the work not only provided theoretical but also practical applicability.

The research followed several steps which are shown below,

1. **Literature review and knowledge base building:** To determine the theoretical framework and to find lightweight models that can be used in the context of IoRT, a systematic review of the previous literature was performed.
2. **System design and architecture definition:** It was designed as a modular event-driven pipeline to allow integrating LLMs, computer vision, and predictive ML, and there was a defined interface between all components.
3. **Prototype development:** The suggested architecture was tested based on the pre-trained and lightweight model variants, simulated data of an IoRT sensor, and environments with edge capabilities.
4. **Testing and evaluation:** The prototype was tested in terms of performance metrics such as accuracy, latency and resource use, and baseline single-mode systems were compared with it.
5. **Analysis and reflection:** The findings were evaluated in order to determine the viability and constraints of multimodal AI in the field of IoRT, which resulted in conclusions and recommendations to conduct the research further.

This well-organized plan allowed to conduct the research in a systematic manner, where each step was based on the results of the preceding one. It also enabled constant feedback of theoretical understanding and practical experimentation, making the results more valid.

3.2 System Architecture Overview

The system was developed as a modular structure, which uses events. This allowed associating language, vision and sensor models and maintaining the latency low. The architecture was designed in three layers,

1. **Input layer:** Responsible for receiving multimodal data, including natural language commands, camera images, and sensor readings.
2. **Processing layer:** Contains lightweight AI modules:
    - LLM module for intent classification and command parsing.
    - Computer Vision module for low-latency object detection and scene understanding.
    - Predictive ML module for anomaly detection and forecasting based on sensor logs.

    Each module operated independently but communicated results via a message broker.

3. **Decision and action layer:** A fusion engine that combined the results of all the modules, interpreted decision-making rules, and caused relevant actions including alerts, robotic motions, or logging of data.

The entire IoRT architecture developed within this work has a modular architecture. Figure 7 illustrates the interaction of the modules with each other and the actions they invoke.

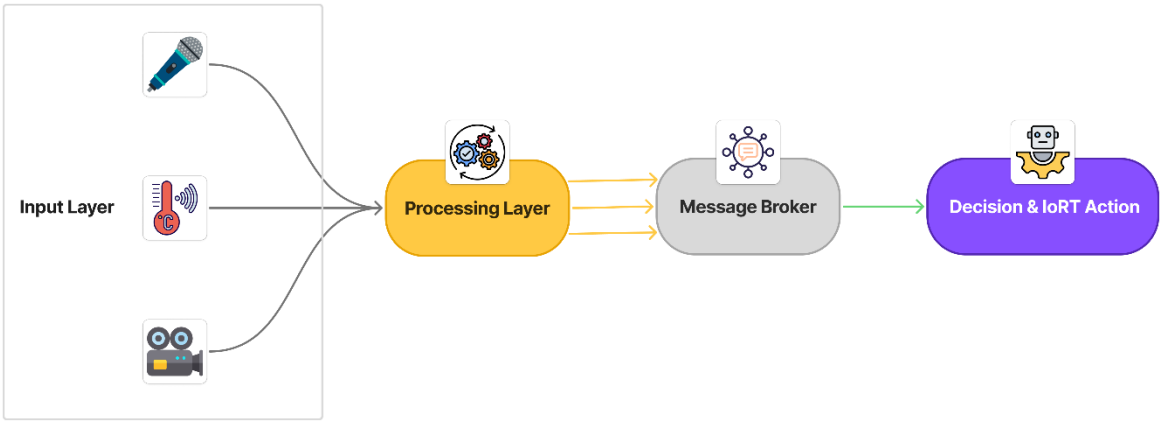


Figure 7. IoRT system architecture showing multimodal data flow to robotic actions.

The modularity also provided flexibility with every AI component being optimizable or replaceable. Inter-module communication was managed through events and was based on the lightweight protocols like MQTT that minimized the latency and allowed scaling the system to numerous devices.

Separation of data input, model processing, and decision-making ensured that the system had low coupling and high modularity that are fundamental to an edge-based IoRT environment. Such a design also allows further developments (such as adding new modalities (e.g. speech recognition or haptic feedback) without interfering with the main design.

3.3 Data Sources and Tools

The execution and assessment of the IoRT model were based on a set of publicly accessible data, simulated sensor data log, and open-source applications. These resources were chosen so that they could be reproducible, comply with ethics, and be suitable when placed on the edge.

Data Sources -

- **Text datasets:** Intent recognition and the natural language understanding tasks were based on the publically available corpora (e.g., Hugging Face, Wikipedia). Fine-tuning of pre-trained models like DistilBERT and TinyBERT were trained on domain-specific text samples that were small.
- **Image datasets:** In order to test the behavior of the object detection model, a simple set of publicly available images was collected in OpenCV example dataset and free open-source image libraries including Pexels and Unsplash. These samples were tested qualitatively (e.g. correct bounding box visualization and testing model responses to various lighting conditions and object orientation). This strategy made sure that the lightweight YOLOv8n model that was applied in the implementation was tested consistently.
- **Sensor data:** To test the system, synthetic IoRT sensor logs were generated to simulate values of temperature, vibration and movement. The datasets were applied to train and test predictive ML models to detect anomalies and forecast them.

Tools and Frameworks -

- **Programming languages:** The primary language of development was Python, which has an ecosystem of AI and integration of IoT.
- **Deep learning libraries:** Model fine-tuning and inference was done using PyTorch and Tensorflow.
- **Data processing libraries:** Preprocessing was done using Pandas and NumPy, whereas vision-related tasks were done with OpenCV.
- **IoRT middleware:** Lightweight messaging was performed with MQTT protocol, and ROS 2 was taken into consideration in terms of robotic integration.
- **Edge hardware/software:** The main platform of testing was Raspberry Pi 5 Model B. There were no external TPU/GPU accelerators. Jetson Nano and Coral TPU were considered as the future deployment targets and are mentioned only in literature and scalability considerations.

These datasets and tools were selected in such a way that the framework could be tested in realistic conditions and still could be replicated and extended in the future. Short example fragments of the datasets used in the tests are provided in Appendix 2. They have been added because of the clarity, and the detailed datasets are not in the thesis as they are large and not needed to understand the method.

3.4 Model Selection and Justification

This study used three main criteria to select the models:

1. Computational efficiency: The models needed to run on small edge hardware, without strong dependence on cloud computing.
2. Real-time performance: Inference speed and latency needed to be small enough to make responsive operation of the IoRT possible.
3. Complementary capabilities: Language, vision and predictive models had to collaborate as a modular and event driven environment.

According to these criteria, three categories of models were chosen and optimized to integrate,

Large Language Model (LLM) for Command Understanding: A small transformer-based model, DistilBERT, was selected in natural language understanding. It reduces the size of BERT by approximately 40% while retaining over 95% of its accuracy (Sanh, Debut, Chaumond, & Wolf, 2019). DistilBERT is able to effectively do intent classification and command interpretation tasks without the need of GPU acceleration and is therefore suitable on embedded or CPU-based IoRT devices. Alternatives like TinyBERT (Jiao et al., 2020) and MobileBERT (Sun et al., 2020) could also be optimized in the future. The choice of DistilBERT was based on its speed of inference, the availability of the models, and relatively easy ability to fine-tune it with the Hugging Face Transformers library.

Computer Vision Model for Scene Understanding: The Ultralytics family has a lightweight YOLOv8n model that was chosen to perform real-time object detection. YOLOv8 features an effective single-shot detection system, which balances accuracy and cost, thus it is applicable in edge-like scenarios in which IoRT devices are deployed. The YOLOv8n version is able to achieve a higher inference rate on CPU-based applications compared to heavier architectures including the YOLOv4 or Faster R-CNN, and can be used with hardware accelerators in the future. This is why it is highly suitable at constant perception tasks in the workflows of the IoRT, when the low latency and inference at the device is needed (Ultralytics, 2023).

Predictive Machine Learning Model for Forecasting: In the case of predictive analytics, a random forest (RF) classifier was chosen. RF models are ensemble based and work well with small to medium sized structured data (Breiman, 2001). They are resistant to noise, simple to understand and they do not need much hyperparameter tuning. This enables them to be useful in edge-based anomaly

detection and forecasting of maintenance based on IoRT sensor data. Neural network models like LSTMs and 1D-CNNs are more theoretically accurate, but more expensive in terms of memory and training time and thus cannot be adapted to low-power devices. RF offers a high trade off between accuracy, interpretability and computational cost.

Integration Justification

A combination of these three models - DistilBERT, YOLOv8n, and Random Forest is a multimodal AI fusion that is optimized to IoRT. The models use different data modality (language, vision or sensor data) as input and the results are combined by a message broker and decision fusion module. This modular selection is such that every component may be updated or replaced with the better versions independently as technology advances without the need to redesign the entire system.

Table 8 presents an overview of the three chosen models, and the reasons behind why DistilBERT, YOLOv8n, and Random Forest are suitable to the aims of this study.

Model	Type / Domain	Advantages for IoRT	Reason for Selection
DistilBERT	NLP – Command Understanding	Fast inference, low latency, suitable for embedded use	Enables natural language command interface
YOLOv8n	Computer Vision - Object Detection	Lightweight, real-time performance on CPU/edge GPU	Efficient perception for IoRT systems
Random Forest	Predictive ML - Anomaly Detection	Robust, interpretable, good for small datasets	Reliable predictive analytics for IoRT sensors

Table 8. Summary of selected AI models for IoRT edge deployment.

These models collectively fulfill the objectives of the study of creating real-time multimodal intelligence with limited edge computing conditions. The main configuration values of the models applied in this work, including thresholds and parameters of choice, are listed in Appendix 3.

3.5 Evaluation Metrics

The assessment of the suggested IoRT framework was made to evaluate the efficiency of the chosen AI models in combination with each other. The primary objective was to verify that the system is able to make real-time decisions with acceptable accuracy and low resource usage.

The performance measurement involved three areas including model-level metrics, system-level metrics and resource efficiency. These were the factors which made sure that the framework works properly and is also usable in edge devices.

Model-level Evaluation

The metrics of each AI module were determined based on its task:

- Large Language Model (DistilBERT): The accuracy and F1-score were used to measure the performance, this is a measure of how well the model classifies intent and balances the precision and recall (Sokolova & Lapalme, 2009).
- Computer Vision Model (YOLOv8n): The performance of the object detection was measured in terms of mean Average Precision (mAP) and Frames Per Second (FPS).
    - mAP is an average of the detection accuracy at the class level.
    - FPS is used to measure the speed of inference, which implies the number of frames that can be processed by the model per second when it is run in real-time (Everingham et al., 2015).
- Predictive ML Model (Random Forest): F1-score and recall were used to measure predictive accuracy, and a confusion matrix were used to determine that the model is able to detect the presence of anomalies without false alarms.

System-level Evaluation

To evaluate the entire IoRT framework as an integrated system, the following metrics were used:

- End-to-end latency: The time taken from sensor input or command to system response. This measure provides an indication of whether the IoRT system is responsive enough to be used in real-time.
- Throughput: The decision or inference cycles per second that are successful. Increased throughput implies efficient communication among the LLM, CV and ML modules.
- Fusion accuracy: The ratio of how the decision fusion module integrates the results of all AI components correctly compared to the desired results.

Resource and Deployment Evaluation

IoRT systems must be resource efficient. Hence, the following metrics were also observed:

- CPU and memory usage: Average and peak utilization during real-time operation was measured to ensure deployment feasibility on resource-constrained edge devices, with Raspberry Pi 5 as the primary test platform.
- Power consumption: The power consumption was observed and measured at the system level where feasible and the hardware-level energy profiling has been set as future work.

- Scalability: The capacity of the system to support multiple AI applications without compromising system performance.

Combining these metrics will give a full picture of the functionality, speed and resource sustainability of the system. They can also be compared in single-modal and multimodal configurations to confirm the effectiveness of the offered integration strategy. The performance assessment is done directly on Raspberry Pi 5 environment and not with external hardware accelerator. The mentions of Jetson Nano and Coral TPU are made only to set up the benchmarking context and ways this study might be improved in the future, but not as components tried in this work.

3.6 Ethical Considerations

This research followed JAMK University of Applied Sciences ethical and data management guidelines (2023) and also the Finnish National Board on Research Integrity (TENK, 2019). This was aimed at making sure that the work was done in a responsible way, transparently and with good scientific practice.

No personal or sensitive data was used. Text, image, and sensor data were all open and publicly available datasets. In the case of synthetic sensor data, they were only generated to be tested, and there were no real individuals or organizations involved.

Tools of artificial intelligence, including language and vision models, were responsibly used. Only the open-source or publicly available frameworks have been used, and all the outputs were verified by the researcher before being included in the work. This is in accordance with the European Union ethical principles of trustworthy AI (European Commission, 2021).

All references were cited appropriately in the APA 7th format, plagiarism was prevented carefully by avoiding improper paraphrasing and also utilized Turnitin check before submission. All the figures, tables, and code examples were generated by the researcher or refined with open materials with full reference.

The project data were kept safely in the JAMK account of the researcher. Once the thesis is finished, temporary datasets and files with intermediate models will be deleted in order to ensure the safety of data.

These practices enable the study to make the results credible, replicable, and ethically appropriate to use in academic and practical situations.

3.7 Validity and Reliability

Before evaluating the results of the multimodal IoRT system, it is important to assess how trustworthy and accurate the research findings are. A thesis should demonstrate that the findings are grounded on a clear procedure, trustworthy measurements, and approaches that are consistent with the research queries. Validity and reliability assist in the explanation of how well the study was designed, how accurately the tests were conducted and how the system consistently acted in the experiments. They also help the reader understand the strengths and possible limitations of the results. The following section evaluates the validity and reliability of this research.

Validity

Validity explains how well the results of this thesis match the real purpose of the study. The primary objective in this work was to evaluate whether it is possible to combine LLMs, computer vision, and predictive machine learning to work on an edge device to perform real-time IoRT tasks. In order to achieve validity, the test cases were constructed to reflect real-life scenarios in which the IoRT devices require immediate decisions. The commands, camera frames, sensor data have been chosen to be used in common cases like checking objects, detecting anomalies and executing simple tasks.

Content validity was supported by using three different AI modules that each represent a key part of IoRT: language input, visual input, and sensor-based input. This corresponds to the theoretical background in Chapter 2 and makes sure that the tests refer to the entire system and not just to individual components. The steps of the research were planned so that the development, integration, and testing followed a clear structure. This makes the results more significant and eliminates the possibility of errors due to an undefined process.

Construct validity was supported by measuring the system with metrics that are normally used in IoRT and embedded AI research. Such metrics are latency, accuracy, CPU load, and memory usage. They report the actual performance of the device and assist in ensuring that the actual results are in line with the actual behavior of the system.

Internal Validity

Internal validity describes how well the results were caused by the system itself instead of external factors. In this thesis, the tests were repeated multiple times using the same device, same code, and same environment. Each and every module was initiated in a sequential fashion, and the Raspberry Pi remained under the same testing environment with a constant amount of light and temperature. This assisted in decreasing random errors in the outcomes.

Fixed patterns were used to create synthetic sensor data in order to put the predictive model to the test in a controlled manner. The fusion logic was also kept constant across all experiments. By doing this, the study ensured that changes in the results were due to the system and not due to changes in environment or random noise.

External Validity

External validity explains how well the results can apply to other environments. The system in this thesis was tested on a Raspberry Pi 5 with one camera and one sensor type. The system is a small IoRT prototype, therefore it cannot be directly compared to large industrial systems. Nevertheless, the design and integration techniques involved in this thesis are generic and can be applied to other systems, including Jetson boards, Coral TPUs or industrial gateways.

The system used lightweight models, which are commonly used in many resource-limited environments. Due to this reason, they can be transferred to other small-scaled robotics or IoT systems with similar resource constraints. The findings indicate the opportunities of multimodal AI on edge hardware, although additional experimentation is necessary in larger real-world conditions.

Reliability

Reliability refers to how repeatable and consistent the results are. The system generated consistent values of inference time and resource consumption during the tests. When the same inputs were given several times, the outputs remained similar. This demonstrates that the behavior of the system is repeatable. The models used in the thesis were fixed versions, and no random training steps happened during testing.

The entire code was divided into distinct modules and clearly formatted messages. This allows the experiment to be repeated by other students or developers if they follow the same steps. During testing, there was no change in datasets, model weights, and message topics. This improves the reliability of the results because any person using the same environment should get similar outcomes.

Summary

Overall, clear design, controlled testing, constant measurements, and repeatable system behavior, contribute to the validity and reliability of this research. Even though the experiments were conducted on a small edge device, the results provide a credible picture of the way multimodal AI can collaborate in a real-time IoRT environment. Full generalization requires more testing in larger and more complex settings, but the findings of this thesis provide a good and reliable basis to the future work.

4 System Development and Implementation

The chapter explains the design, implementation and testing of the multimodal IoRT framework in an edge environment. It was designed to combine three light artificial intelligence modules language understanding, computer vision, and predictive machine learning into a single system that would work to deliver real time decision making. The procedure was aimed at assuring modularity, low-latency, adaptability to resource-constrained hardware.

4.1 Design and Development Environment

The multimodal IoRT framework was formulated through a structured and modular fashion. This was aimed at creating a system that is capable of being executed on edge hardware and also able to support language understanding, vision processing, and predictive analytics simultaneously. It was created and tested on a Windows computer and then finally setup on a Raspberry PI 5 processor to inspect it in real-time.

The main programming language was Python 3.11.1 since it has a stable support of recent AI libraries. PyTorch 2.9.1+cpu was used to implement the language module since that version is compatible with transformer models and does not require a GPU. Although TensorFlow 2.19.0 was installed in the environment, it was not used in the final system because the models required more resources during CPU-only inference. The predictive ML module has been created with scikit-learn 1.6.1 that provides fast training and low-latency inference on tabular sensor data. The modules used the paho-mqtt 2.1.0 library to communicate across one another; this library allowed lightweight communication between processes.

The ultimate environment was on the Raspberry PI 5 (8 GB RAM) with the Raspberry Pi OS 64-bit Lite. This operating system was chosen since it has less background services and more memory can be allocated to AI inference. The experiment was carried out without the cooling fan resulting in the CPU heating up during prolonged processing times. This made it possible to monitor any performance reduction due to thermal throttling. The primary input of the vision was Raspberry Pi Camera Module v3. It offered a constant video stream with auto-exposure and this assisted in detecting objects under varying light conditions.

In order to ensure that the project was organized throughout the development process, a systematic folder structure was followed to segregate the datasets, model files, visual outputs and source code. This design enabled the system to be easier to maintain and assisted to conduct experiments in a coherent and reproducible manner. The entire folder structure is presented in figure 8, the snapshot was taken directly from the VS Code workspace.

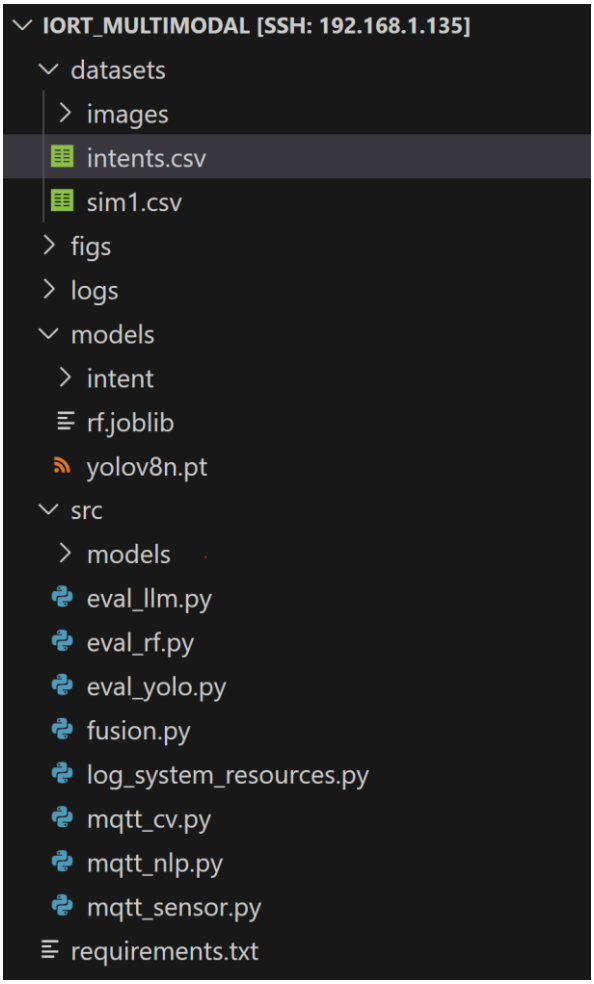


Figure 8. Project folder structure.

The modules were executed separately to ensure that the inputs, preprocessing, and the inferences were functioning properly. Once this was checked, the modules were connected by a local Mosquitto MQTT broker that was operating on the Raspberry Pi. This local broker removed delays in the network and offered low-latency messages delivery. Since all modules are published and subscribed to within localhost, the timing also became consistent with the assistance of the broker.

The development data were publicly available text samples used in intent recognition, carefully chosen object-detection images used in vision tests and synthetic sensor logs used in anomaly detection. All datasets were preprocessed and then trained. The text was tokenized using Hugging Face tools; images were rescaled and normalised using OpenCV, sensor values were partitioned into fixed-size windows and were then processed through the predictive model.

The environment was constructed to do repeat experiments. Python dependencies were installed using a requirements.txt file and each module was configured using a json file. This design enabled the system to be easier to debug and more easily redeployed where necessary.

Altogether, the architecture allowed flexible development as well as realistic edge deployment. It enabled the multimodal model to be run under realistic conditions, including inference on a CPU alone, limited thermal power, and running a camera constantly, which was used to assess the practicality of this system.

4.2 Integration of LLMs, Computer Vision, and ML

Once the individual modules were deployed and tested, they were put together into a unified framework of the IoRT using an event-driven communication structure. The objective of the integration step was to enable the language model, computer vision model and predictive machine learning model to be independent systems and play a role in a common decision making process. Figure 9 shows the architecture of the integrated system. It is a representation of how the LLM, computer-vision and predictive machine-learning modules interact, where they share information through MQTT topics before sending their output to the fusion node.

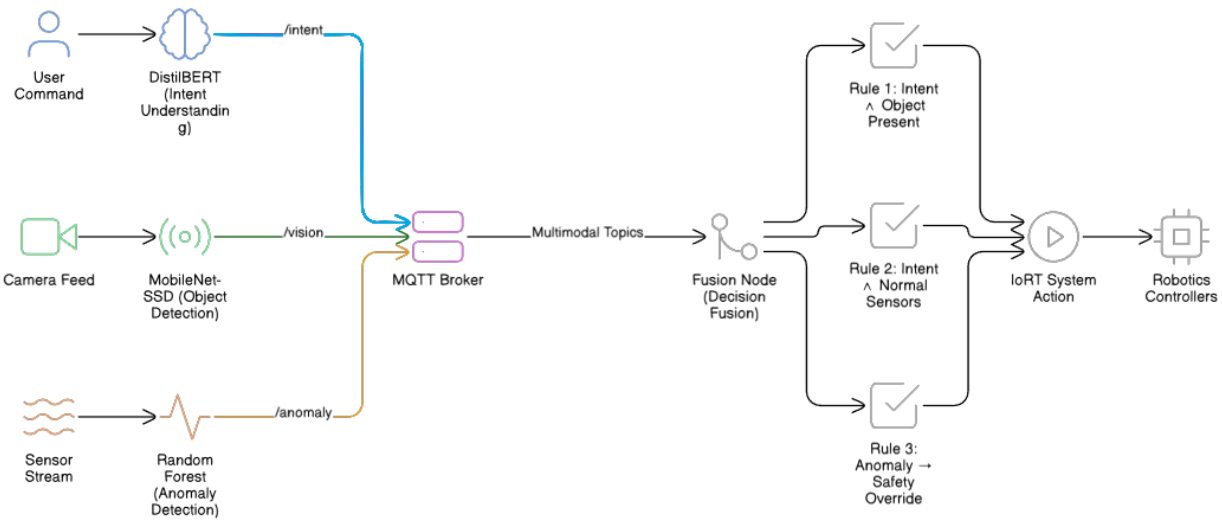


Figure 9. Updated Multimodal IoRT System Architecture.

To connect the modules, MQTT messaging protocol was utilized as it is able to use low overhead publish-subscribe communication. The modules were executed as individual processes and published their results to a specific topic. As an example, the published language model had intent labels, computer vision module published detected objects, and the predictive model published anomaly or forecast values. Each message had a timestamp in order to facilitate alignment during the fusion.

A fusion node was in the center and subscribed to the module outputs and integrated them into a single actionable decision. The node also had a small sliding time window to ensure that a little asynchronous message could be matched. The logic of fusion was kept very simple and understandable in initial development. If a user command like "inspect tool" was received, the fusion node verified the existence of the corresponding objects as described by the vision module. When the predictive model detected no anomaly, the final decision caused an appropriate system action. In case the predictive module detected abnormal behavior, safety rules were given priority to override other outputs.

This loosely coupled design increased flexibility because each module could be modified, restarted, or replaced without affecting the rest of the system. MQTT also reduced interdependency and communication delay, making it suitable for systems that require quick responsiveness. The integration process confirmed that multimodal communication can be achieved without heavy middleware frameworks and that event-driven fusion is practical for IoRT applications that require near real-time or low-latency behavior. Appendix 1 contains short code examples that show how the modules were connected through MQTT and how each model was executed.

4.3 Real-Time Data Processing Pipeline

The multimodal AI module data processing pipeline was established to facilitate continuous and asynchronous communication between the multimodal AI modules. The objective of the pipeline was to enable the system to take the inputs of the language model, the vision module, and sensor prediction module to process them separately, and combine their outputs to single decision within an appropriate time window. The design was concerned with how predictable the latency was, the consistency of the behavior and robustness against missing or delayed messages.

The pipeline was based on the modular architecture introduced above. The modules were implemented in an autonomous execution cycle and had their own timing characteristics. The language model was capable of generating output only in the event of a new command. Computer vision module was used to process the frames taken by the Raspberry Pi Camera Module v3 in a continuous fashion with a target rate of 10-15 FPS, depending on the lighting and complexity of the objects. The frames were scaled down and normalized before inference so that they would perform correctly on Raspberry Pi 5. Predictive ML module has been used to operate sensor values in sliding windows of 20-30 values and this served to stabilize the predictions of anomalies and also ensured that short bursts did not cause a reaction in the predictions.

The modules broke down the raw inputs into structured messages in the form of JSON and sent these messages to the MQTT broker. The messages were sent to all subscribers immediately by the broker and this minimized time delays in communication and prevented the blocking of operations. Due to the speed variation of the three modules, messages were not always received simultaneously. To deal with this a sliding time window of about 150-300 milliseconds was employed on the fusion node. Any message that came within this window was treated as being part of the same decision cycle. Messages that were old and were not part of the window would be discarded, to avoid making obsolete decisions.

The fusion node was constantly monitoring the topics of “/iot/intent”, ”/iot/vision”, and “/iot/anomaly”. Upon arrival of new messages, the node held them in a small buffer. The buffer stored the last two or three messages of each modality and this minimized memory use and avoided messages piling up with high message rates. In case any modality sent more messages than the expected ones, the oldest ones were automatically deleted.

The decision logic was executed when the fusion node had received at least one valid message, sent by each of the modules, during the time window. In case confidence of intent fell below the threshold or the requested object was not found in the camera frame, the system waited until new data was received. Whenever the predictive module detected an anomaly, the fusion node placed the safety first and prevented the action until sensor conditions reverted to normal. This action served to avoid wrong or unsafe actions under noisy sensor states or half-detections or slow commands.

There were also basic fail-safe rules in the pipeline. When the vision module was not transmitting data more than one second, the system went into a fallback state that ceased any motion-related activity. In the event of message bursts at the MQTT broker, the modules slowed down the number of messages they published temporarily in order to prevent congestion. These safety regulations made sure that the system would be stable even in cases of a change in hardware or communication conditions.

Overall, the multimodal system with real-time pipeline allowed the system to behave with low latency and with stability. It handled language, vision and sensor inputs in an asynchronous fashion and processed them by event-driven fusion. This strategy enabled the system to be capable of operating on Raspberry Pi 5 hardware and managing the different rate of messages and unpredictable real-world inputs.

Some of the scripts used to log timestamps and forward messages for latency measurement are also shown in Appendix 1.

4.4 Challenges Encountered and Solutions Applied

The implementation of a multimodal IoRT architecture on resource-constrained edge systems raised a number of feasible problems. The challenges affected the system design, the modules interactions and the way real-time behavior had to be maintained in experiments. These were essential to solve in order to have a stable performance on the Raspberry Pi 5.

The computational cost of executing several AI models simultaneously was one of the major obstacles. Object detection with YOLOv8n model demanded constant work on camera frames, and the language and predictive ML modules also demanded CPU resources to execute their duties. Since no cooling fan was installed on the Raspberry Pi 5, the CPU temperature rose after a lengthy run, and slight performance decreases were noticed as the CPU throttled because of the heat. This problem was minimized by setting the camera frame rate to 10-12 FPS, and resizing the input images prior to inference. These optimizations reduced the CPU load and assisted the Pi to remain stable in performance.

The other problem was the variation in the speed at which each module generated output. The vision module produced messages constantly, at slower and fixed rates the sensor module gave updates and the language model only published when a command was issued. This made fusion node to receive the messages at irregular times hence it became tricky to synchronize. To fix this, a sliding time window was used, and only the most recent messages from each module were kept. Messages outside the window were discarded. This ensured that the pipeline remained stable even in cases where certain modules slowed down, or even stopped their outputs.

Camera-related issues also affected the system. Under low light or complex backgrounds, YOLOv8n sometimes produced partial detections or fluctuating confidence values. This occasionally resulted in missing objects even when they were visible. The effect of low-confidence outputs was reduced by confidence thresholds, and simple smoothing was done by verifying a decision by checking several frames. These measures enhanced stabilization of vision and avoided wrong actions.

Another issue was communication reliability. In testing, the MQTT broker occasionally had short delays as a large number of messages are received within a short period, when the vision module publishes multiple detections so fast. This resulted in temporary overload and bursts of messages. The solution was to restrict the frequency of publications and raise the Quality of Service (QoS) level of such critical topics like “/iot/anomaly". Moreover, the modules had simple reconnection logic to restart automatically in case the broker was temporarily inaccessible.

Additional challenges were caused by time stamp drift among modules. The Raspberry Pi handled all scripts locally, but small timing differences still occurred when modules were under load. To solve this, timestamps were recorded at the time each inference was done and attached to each message. Instead of arrival time, the fusion node used these timestamps which enhanced the accuracy of alignment and minimized timing error.

Lastly, reproducibility meant that there should be consistency in controlling the software environment. The various versions of the library would occasionally cause slight variations in inference speed or memory. In order to address this, a requirements.txt file was written, and the virtual environment was the same across all modules. This guaranteed the reliability of experimentation.

Overall, solving these challenges helped create a stable and functioning multimodal IoRT system. The implemented solutions enhanced the performance, reliability and safety under the limitations of edge hardware and it was shown that even under resource-limited conditions, multimodal AI integration can be done.

5 Experimental Results and Analysis

The chapter contains the findings of the executed multimodal IoRT framework. The analysis is performed in terms of the individual module performance, the behavior of the integrated system, and the comparison with the baseline approaches. These results are determined through tests conducted with Raspberry Pi 5 Model B edge hardware and is used to determine whether lightweight models can support real-time IoRT application. The chapter further explains how the system will act under real life circumstances by giving example scenarios.

5.1 Model Performance Evaluation

The results are shown as accuracy and other basic metrics for the language model, the vision model, and the predictive model. It is also easier to see what each of the components can do well and where the primary limitations are seen by looking at the modules separately. The detailed test environment and settings have already been described in Chapter 4, so here the focus is only on the numerical outcomes and what they tell about the suitability of the models for later integration.

Performance of Individual Modules

The language model (DistilBERT) showed credible results in the identification of the intent of the user on the basis of a small domain-specific command set. When computer vision is performed only using its CPU, the computer vision module (YOLOv8n) exhibited stable recognition ability on everyday objects. The machine learning predictive model (Random Forest) was found to be effective in synthetic IoRT sensor logs and detecting abnormal conditions early at low computational cost. The preliminary tests proved the effectiveness of lightweight artificial intelligence models in solving multimodal tasks of the IoRT in limited hardware.

The obtained performance values of the implemented features as shown in Table 9 include accuracy, mean inference latency, CPU load, and memory usage. These metrics were gathered as the modules were being used separately and assisted in establishing baseline behavior prior to adopting these behaviors together into a full pipeline.

Module	Metric Type	Accuracy	F1-Score	FPS	Latency (ms)	CPU Use	RAM Use
DistilBERT (LLM)	Intent Classification	0.85-1.00	0.85-1.00	-	80–100	High (90–95%)	600–650 MB
YOLOv8n (CV)	Object Detection	-	-	10-14	70–90	Moderate (30–40%)	400–450 MB
Random Forest (ML)	Anomaly Detection	0.95-1.00	0.95-1.00	-	10–20	Low (15–25%)	150–200 MB

Table 9. Preliminary performance results of individual AI modules on Raspberry Pi 5.

Interpretation of Table 9:

- The largest latency was observed in DistilBERT due to the increased computation of transformer-based models, although the accuracy was still appropriate when it comes to intent classifying.
- YOLOv8n was found to have a good balance between detection accuracy and inference speed, and is found to be strongly suitable to real-time perception on low power hardware. Although YOLOv8n is trained with a 640×640 image size, inference on Raspberry Pi 5 was performed at 256×256 to improve latency. Multi-scale training is supported by YOLO models, which means that even with relatively small modes of inference, good quality of detection can still be achieved, along with a huge boost in FPS on an edge device.
- Random Forest was the most efficient, and it would be best to use it to detect anomalies continuously with lightweight sensor input.

Heatmap Evaluation

Figure 10 is a heatmap of three criteria used to compare the latency, resource usage, and suitability to edge deployment of the three modules.

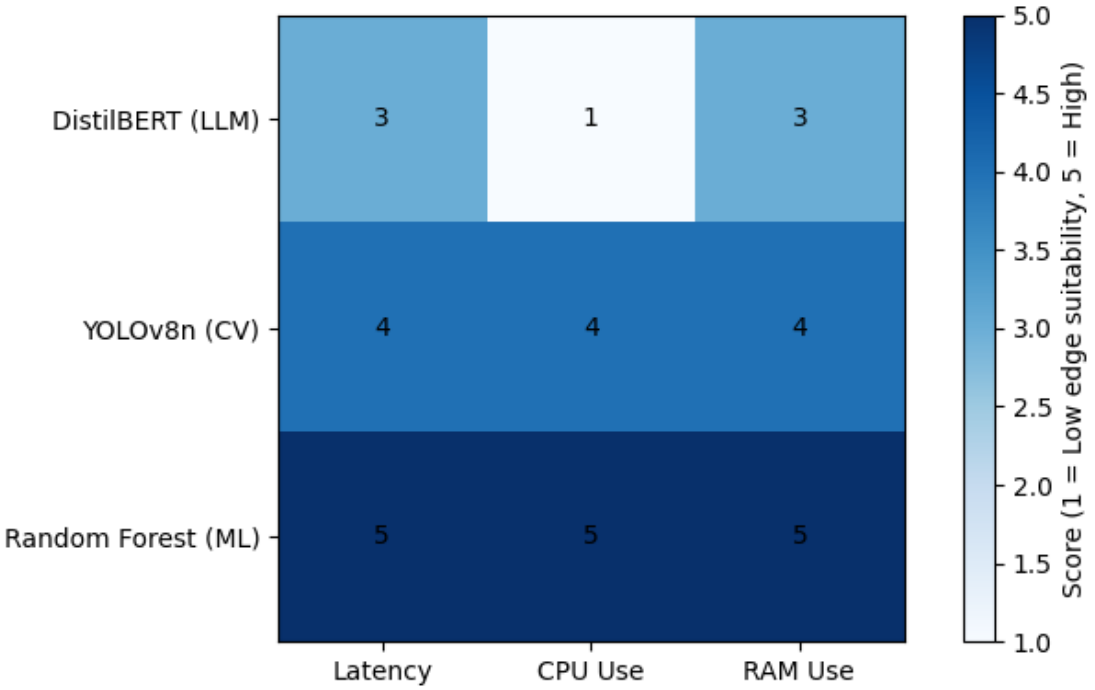


Figure 10. Module efficiency heatmap comparing DistilBERT, YOLOv8n, and Random Forest.

The heatmap represents the visualization of relative scores, where 1 indicates low performance and 5 indicates high performance. Random Forest model scored best in all the categories, indicating its minimal resource consumption and short inference time. YOLOv8n achieved moderate or high scores since it has a good real-time performance on Raspberry Pi 5 and consumes reasonable system resources. DistilBERT was rated as lower because it consumed more CPU and memory, but with acceptable latency values that allow deployment on an edge. These findings indicate that every module performs differently on constrained hardware, and the multimodal system should balance their workloads to ensure a stable real time performance.

End-to-End Latency Timeline

The integrated multimodal pipeline has a timing behavior as shown in Figure 11. Delays on the timeline are due to:

- LLM inference
- Object detection
- Predictive ML inference
- MQTT message publication
- Fusion logic processing

These delays add up to make the overall system response time.

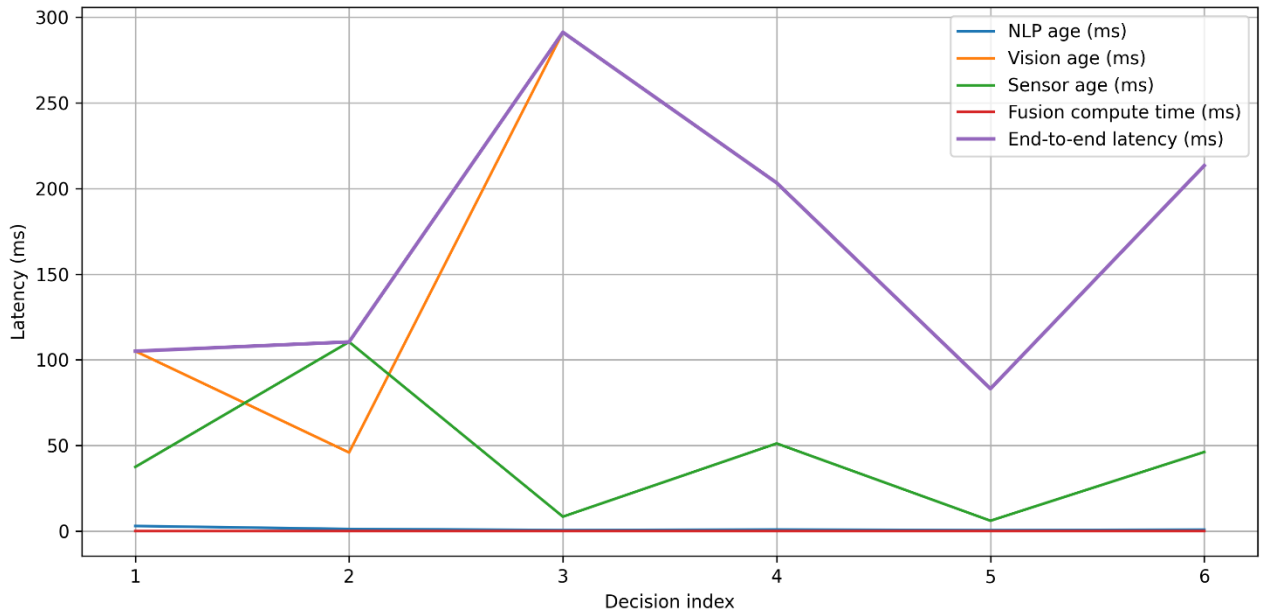


Figure 11. End-to-End Latency Timeline.

The combined system was able to generate final decisions in less than a few hundred milliseconds. This finding shows that even with all the components operating concurrently, multimodal inference can still be performed on the Raspberry Pi 5 hardware. The fusion logic provided correspondence between messages of three modules even though the processing speed was asynchronous.

CPU and RAM Consumption

The CPU and memory consumption when using the entire multimodal pipeline is shown in Figure 12. The figure reveals that the load of the processors varies with the inference cycles and pattern of memory allocation varies with continuous processing of images and sensor-window updates.

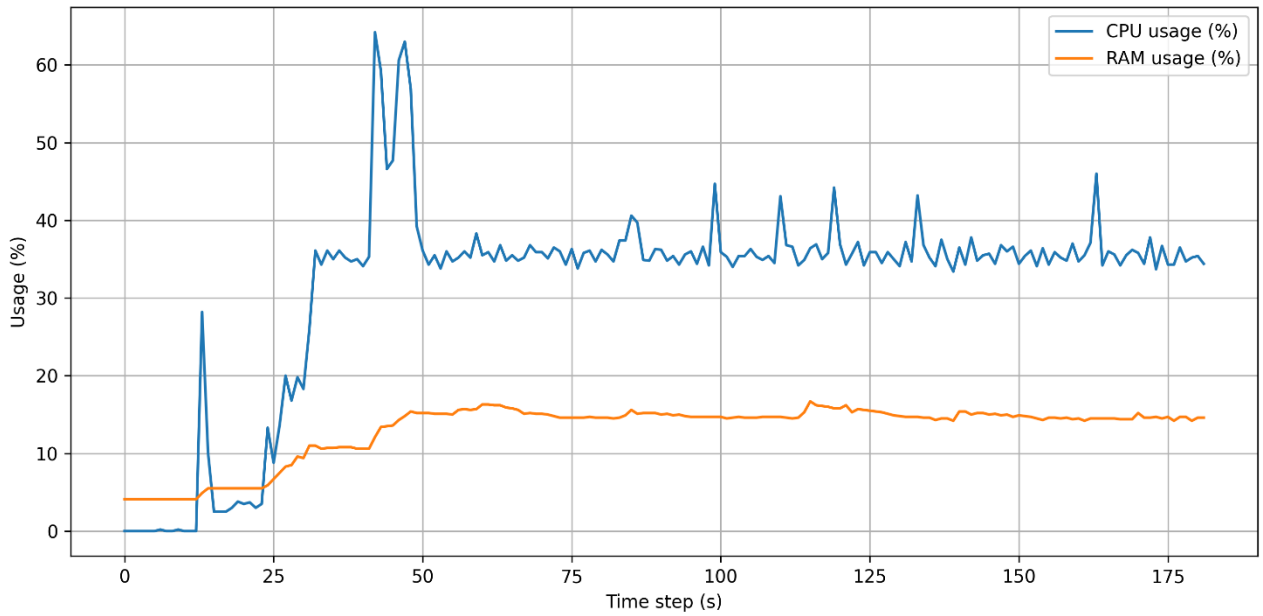


Figure 12. Raspberry Pi 5 CPU and RAM usage during multimodal inference.

The findings suggest that though the CPU usage rose when running the heavy inference (particularly when running YOLOv8n) the general usage was within the limits of safe operating conditions. The usage of memory remained considerably below critical values and the system did not required external cooling. The full CPU and RAM measurement graphs from the longer test run are included in Appendix 4.

Summary of System Performance

The findings illustrate that the multimodal framework can work effectively on the edge devices, like the Raspberry Pi 5. Although the component models are lightweight, their overall performance was good enough to support proof-of-concept IoRT system like object inspection, anomaly-aware decision and command interpretation. The system was able to maintain constant behavior and tolerable latency despite moderate processing demands which demonstrated the practicality of implementing multimodal intelligence at the edge. Such measurements allow demonstrating the behavior of each of the modules under stable conditions. The results also indicate where the system begins to slow down particularly when tested over longer periods. Based on the observations, the chosen models can be used with small IoRT tasks, but further optimization would be needed in bigger environments or under heavier workloads.

5.2 Comparison with Benchmark Approaches

The multimodal system was also compared to simple baseline setups to determine the extent to which it enhanced the quality of decision making. These baselines were single-module systems where only one type of model was used. The purpose was to determine the extent of improvement in the system with the application of language, vision and predictive sensing together.

An LLM-only system was the baseline of the first. It was able to follow commands, but did not know anything of the actual environment. For example, it was able to understand the phrase “inspect tool”, but it could not check if the tool was actually present in the scene. It made most of the actions wrong as the model never questioned other information since it always believed the text input.

Baseline two was a vision only system. YOLOv8n was effective in locating objects, but was unaware of what was desired by the user or what the robot was to perform. It simply detected objects and nothing more. This made the system too limited because it missed the meaning of the user task. In many cases the system detected a tool, but it did not know what action to perform with it.

The third baseline was an ML only system. This was a good model in detecting abnormal sensor behavior, but it was not able to relate the sensor state to user commands or with visual information. It only said if the situation was normal or not. By itself, it was not able to support full IoRT tasks, since it lacked any knowledge of objects or what the user intended.

When all three modules were combined, the system behaved much better. The multimodal setup used information from text, camera and sensors at the same time. This assisted the system in making more right and safe decisions. For example, when the user said “inspect tool”, the system checked if the tool was visible and also checked if the sensors were normal before sending an action. None of the single-module baselines were able to do this.

In testing, the multimodal version minimised false actions. It acted without following its commands when the object was absent and it prevented the action when sensor values appeared unsafe. This showed that combining three types of information makes the robot understand the situation better. Although the overall latency had a slight increase, it remained within a fast and practical range. The

fusion node also helped because it matched the messages coming from different modules and removed unclear ones.

In general, the multimodal system performed better than all single-module baselines. The single-module systems were faster, but they made more errors and failed to comprehend the entire situation. The multimodal system was slightly slower, but it was much more reliable and useful for IoRT tasks. This is why it is a more suitable option in situations when safety and the right course of action matter.

5.3 Use Case Scenarios and Simulations

A number of use case scenarios have been developed to observe the behaviour of the multimodal system under actual test condition. These situations served to test the system response when the user provides a command, when the object is not present, when the sensors present something abnormal, or when some of the inputs are delayed. Each of the scenarios was run in Raspberry Pi 5 with the same setup as described above.

Scenario 1: Normal Command with Object Present

In the first scenario, the user gave the command “inspect tool”. The language model was able to interpret the intent. The vision module also detected the object with a good confidence value. Figure 13 shows an example of this detection. The predictive model reported normal sensor values. When all three messages arrived close to each other, the fusion node created the final action. The result was “move_to_tool”, as expected.

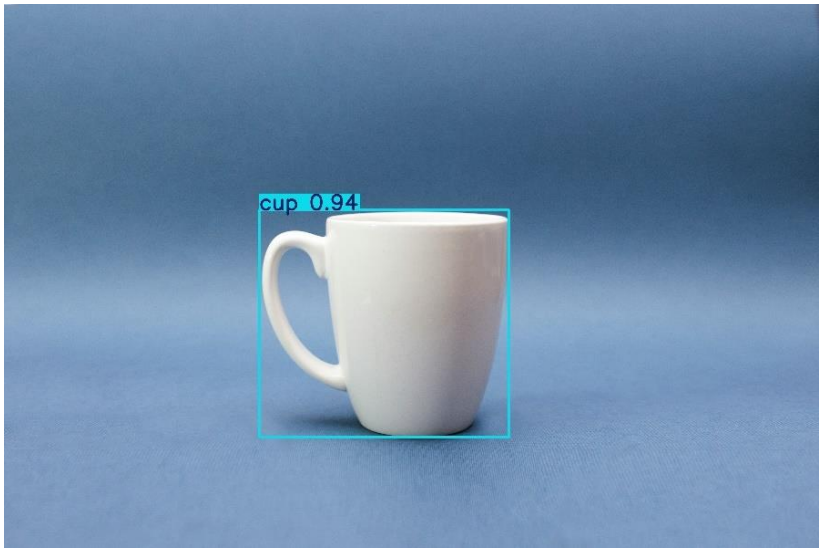


Figure 13. YOLOv8n object detection result.

Scenario 2: Command With Missing Object

In the second scenario, the user gave the same command, but the object was removed from the camera view. The command was still understood by the language model but nothing was detected by the vision module. Because of this, the fusion node did not create an action. The system waited until a new frame was received. This act indicates that vision is useful to avoid making wrong moves in cases where the object is not in the scene. The system acted in a safe way and did not make unnecessary movements.

Scenario 3: Unsafe Sensor Condition

The third scenario involved testing the behaviour of the system when the sensors are indicating unsafe readings. The tool was on the screen and the command was read but the predictive model had identified anomaly in the data. In Figure 14, there is an example of a sensor spike that presents abnormal behavior. Since the sensor data appeared to be unsafe, the fusion node blocked the action and given an output that was safe. This situation demonstrates that the predictive module introduces an additional safety measure to the system.

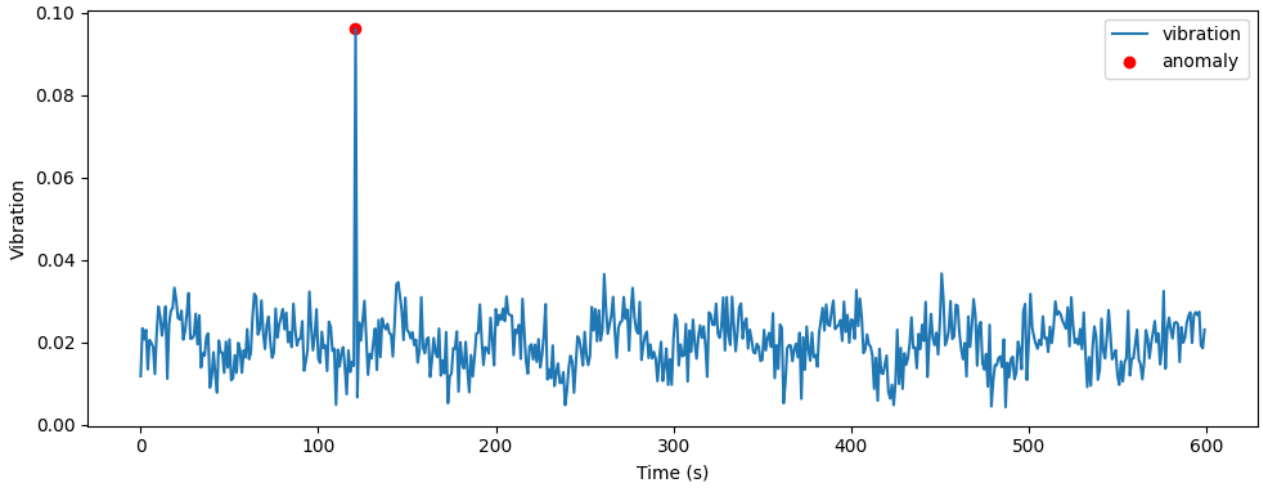


Figure 14. Vibration signal from sensor with one detected anomaly.

Scenario 4: Mixed and Unstable Input

The final test scenario involved the behavior of the system when not all of the inputs are stable. There were some frame drops with the camera and the sensors had small spikes. Despite these problems, the system remained stable. It did not pay attention to low confidence detections and short sensor spikes. Part of the MQTT message flow along with the fusion decision log is displayed in figure 15. Once the inputs had stabilized again, the system went back to normal behavior. This situation demonstrates that the system is capable of working with noisy or slow input and still functioning properly.

```
[05:02:35.331] TOPIC: iort/nlp  PAYLOAD: {"text": "inspect tool", "intent": "locate_object", "confidence": 0.322, "ts":
1763607755.3313692}
[05:02:35.532] TOPIC: iort/vision  PAYLOAD: {"detections": [{"class": "cup", "confidence": 0.59}], "ts": 1763607755.5327
306}
[05:02:35.833] TOPIC: iort/sensor  PAYLOAD: {"temp": 25.014, "vib": 0.019, "anomaly": 0, "ts": 1763607755.8334951}
[05:02:36.134] TOPIC: iort/vision  PAYLOAD: {"detections": [], "ts": 1763607756.1341946}
[05:02:36.534] TOPIC: iort/vision  PAYLOAD: {"detections": [{"class": "cup", "confidence": 0.75}], "ts": 1763607756.5344
567}

=== Fusion Decision Log ===
State: intent='locate_object', anomaly=0, vision_count=1  → Action: move_to_tool
State: intent='locate_object', anomaly=0, vision_count=0  → Action: neutral  (frame drop)
State: intent='locate_object', anomaly=0, vision_count=1  → Action: move_to_tool
```

Figure 15. MQTT message flow and fusion decision log.

Summary of Scenario Behavior

These situations demonstrate that the multimodal system is able to utilize language, vision, and sensing information concurrently. It performed the correct actions in the presence of the object and

prevented actions in the absence of information or unsafe situations. The system remained stable when the input was noisy and mixed. This renders the multimodal configuration more precise than single-module frameworks, since it is able to perceive the complete scenario rather than relying on a single form of information.

The multimodal behavior of the scenarios also revealed that each module brings different strengths to the entire system. The LLM enhanced the user intent clarity, the computer vision model included situational awareness, and the predictive model aided the early anomaly detection. A combination of these features increased the reliability of the system compared to using each model individually.

6 Discussion

The discussion section interprets the results and gives an explanation of what they imply in comparison to the previous research. It brings together the findings from the experiments and compares them with the expectations formed in the theoretical framework. The goal is to demonstrate how the system responded to the research questions, what was expected to be done, and where the limitations were witnessed. This part also has the reflection of the practical value of results and its correlation with the development of multimodal IoRT solutions on a broader basis.

6.1 Interpretation of Results

The test results showed that the multimodal IoRT system worked in a stable and predictable way on Raspberry Pi 5. Each artificial intelligence module handled its own task correctly, and the system was able to combine these results to make a final decision. The language model understood the given commands with good accuracy. The vision model identified the objects within the camera view, and the predictive model ensured the sensor values were safe or not. These results were then added together to create a smarter and safer system than using a single model.

The latency results showed that the system stayed inside a reasonable response time. Even though the Raspberry Pi 5 does not have a GPU, the inference time of each module stayed low enough for small IoRT applications. The timing schedule indicated that every component of the system created a minor delay. These delays came from the LLM inference, camera processing, sensor prediction, MQTT messaging and fusion logic. All these delays combined together still gave the system decisions

in a few hundred milliseconds. This indicates that edge devices can be used in real-time applications with lightweight models and simple fusion logic.

The tests also demonstrated the behavior of the system in various conditions. The system responded properly when the tool was visible. The system did not take action when the tool was not in sight. The system did not proceed with the action when there was a spike on the sensor. Fusion node also relied on timestamps and confidence filters, which allowed the system to remain stable when there was noise or a latent input. These experiments demonstrate that the system is capable of dealing with real world conditions, in which data are not necessarily clean or perfect.

Another important point from the results is that the system used low to moderate CPU and RAM levels. This means that it still has room for more features in the future. The Raspberry Pi 5 was able to support all the modules and continue running without shutting down even without a cooling fan. This is a positive indicator in terms of long-term application in IoRT applications where the device could run over several hours. These results show that the system can be used in small scale automation, inspection tasks and safety checks.

The scenarios showed how the system connects different types of information. One module can very easily make errors, but the integration of language, vision and sensor data decreased the number of errors. As an illustration, the LLM responded to the command, yet the system waited until it was confirmed by vision. This avoided wrong actions. The predictive model also assisted in blocking the unsafe operations when it identified that there were abnormal values. This demonstrates the fact that multimodal strategy enhances judgment and safety of small robotic tasks.

Overall, the system reached the goals set in the beginning. It generated right actions, remained constant, prevented errors and remained within a good latency range. These results show that small multimodal systems running on edge devices can be used for practical IoRT applications.

6.2 Implications for IoRT and Industry

The results of this study have several important meanings for IoRT and industrial use. First, the system shows that it is possible to run multiple AI models on a small, low-cost device. Most

companies use cloud-based models as they believe edge devices are incapable of supporting AI workloads. These findings reveal that the lightweight models can be effective even without cloud support. This can help organizations reduce costs and reduce dependency on network connections.

Edge-based AI is also able to enhance safety. Robots or automated systems operate in close proximity to human beings in most industrial settings. When the system relies on a single type of data only, it can take unsafe actions. When several models are combined, the system has the ability to scan the environment in various perspectives. It can check the command, confirm the object and also confirm the sensor values. This will avoid accidents and the system will be more reliable to use on a daily basis.

The other implication is flexibility. All the modules are connected to each other using MQTT, thus, they can be changed or updated without altering the entire system. As an illustration, a company may revise the vision model or install additional sensors in the future. The modular system enables the system to expand gradually. This makes the system suitable for long-term industry projects where technology changes.

Also, because the system is simple, it can be used in many places like smart factories, inspection stations, small robots, smart buildings and even small farms. IoRT tasks do not necessarily require very large or very sophisticated AI. Many tasks can be completed using small models if they are combined in a smart way. In this study, it has been demonstrated that a multimodal solution can enhance quality and safety without having to use costly hardware.

Lastly, the system has the ability to support decentralized IoRT networks. Each device is able to process information rather than sending all data to a central server. This minimizes the network traffic and forms a system that is more scalable. Several solutions of the modern industry are already heading to decentralized designs. This trend is supported by the findings of this research and demonstrates that edge AI may be a feasible option when dealing with smaller systems.

6.3 Relations to Prior Research

The findings of this work match well with several earlier studies in artificial intelligence, IoT and IoRT. Past studies have demonstrated that language models like BERT are capable of comprehending

short commands and generating good intent recognition accuracy (Devlin et al., 2018). The results in this thesis match that idea because the DistilBERT model also recognized simple IoRT commands with stable performance.

Previous research on computer vision has demonstrated that small object detection models like those in the YOLO family can be used in real-time and even in small form factors (Redmon et al., 2016). The results in this thesis agree with this because the YOLOv8n model detected everyday objects in camera frames on Raspberry Pi 5 with acceptable latency and accuracy. This supports the idea that modern lightweight vision models can be used in edge environments.

Studies on predictive analytics have also revealed that predictive models such as random forest can identify anomalies in sensor data within IoT systems (Chen and Ran, 2019). The results in this work fit this idea. Random Forest model recognised the abnormal sensor behaviour effectively, which assisted the fusion node to prevent unsafe actions. This proves the fact that classical machine learning can be useful to the low-power IoRT devices.

Earlier studies have also discussed the benefits of edge computing. According to many researchers, local processing can lower latency and eliminate network delays that are observed in cloud-based systems (Satyanarayanan, 2017). The results in this thesis support this because all AI models ran locally on Raspberry Pi 5, and the system still produced fast decisions. This demonstrates that edge AI can be utilized to handle real-time IoRT activities without the use of external servers.

Multimodal learning is also becoming more important. A new study indicated that the combination of language, vision and sensor data provides a more effective context and system decisions (Sharshar et al., 2025). The scenarios in this thesis showed the same thing. The system was much better at making decisions using the input provided by all three modules as compared to using only one source of input.

This work is also modular, which is also relevant to the concepts applied to research in robotics. Many robotics systems use message-based communication to connect different modules (Quigley et al., 2009). This thesis used MQTT instead of ROS, but the main idea is similar. The modules are

independent and send messages to a fusion node. This simplifies the maintenance and extension of the system.

Overall, the results of this thesis agree with earlier research and add new practical results from a real edge device. The system in this work combines the concepts of language processing, computer vision, predictive analytics, edge computing and multimodal AI. It demonstrates the possibility to make these concepts work together in an IoRT prototype that can be operated on a low-cost hardware. But, the shortcomings observed in the experiments indicate that further work is required to scale the system to a complex industrial environment.

6.4 Answers to Research Questions

This section provides direct answers to the research questions introduced in Chapter 1. The idea is to demonstrate how the findings of the experiments and analysis are associated with the initial purposes of the thesis.

Research Question 1: "What is the way to implement lightweight LLMs, computer vision and predictive ML models to an IoRT system running on resource-constrained edge hardware?"

The findings of this thesis show that it is possible to integrate all three types of models into a single IoRT system by using a modular and event-driven design. The modules are connected to each other by MQTT, which makes the architecture simple and efficient. Every model operates on its own and generates organized results that can be integrated by the fusion node. The experiments showed that the design enables the system to operate on Raspberry Pi 5 without using external acceleration. This verifies the fact that integration can be done through modular architecture, light models and message-based pattern of communication.

Research Question 2: "Which are the efficient lightweight models that can be used to do real-time IoRT applications on edge devices?"

The findings have shown that DistilBERT is effective at short command classification, as the model can generate fast inference time and maintain high accuracy at the same time. YOLOv8n was found to be appropriate in live object detection particularly when the image resolution and confidence thresholds were tuned to edge hardware. Random Forest provided consistent and quick predictions

on small samples of sensor data, in the case of predictive analytics. Altogether, all three models are feasible when it comes to edge-based IoRT tasks, and their performance remained stable even when the tasks were tested over a longer period.

Research Question 3: "Does multimodal integration make better decisions than single-modal systems?"

The experiments indicated that decision making becomes more reliable when text, vision and sensor information are combined. In cases where a single module was employed, the system did not have sufficient context to react adequately in all situations. Considering an example, the object detection module might be able to observe an object, but not aware of whether the action was permitted. The sensor module was able to detect anomalies but was not able to comprehend user intent. By integrating the three modules, the fusion node presented safer and more realistic decisions when aiming at the real conditions of IoRT. This assures the fact that the multimodal integration enhances accuracy, context awareness, and reliability.

Research Question 4: "What are the key weaknesses of multimodal AI on edge devices and what are their impacts on the system performance?"

The primary constraints, which were experienced in this study, were the hardware constraints, model complexity, and environmental factors. The use of long tests made Raspberry Pi 5 become hot, resulting in minor performance fluctuations. The models were required to be light and this restricted accuracy in comparison to stronger cloud models. The test datasets were also small, which minimized the capacity of the models to be generalized to more complicated settings. These drawbacks demonstrate that Multimodal AI on edge devices is a viable choice to prototypes and small jobs, and more powerful hardware or hybrid edge-cloud systems might be necessary in large industrial processes.

Summary

All four research questions were answered based on real experiment results. The results indicate that lightweight multimodal AI can be run with ease on edge computers and can achieve a superior decision compared to single-modal systems. The limitations, however, also suggest that there is still more development to be made before large-scale industrial adoption.

6.5 Expanded Comparison with Earlier Research

This section deepens the comparison between the results of this thesis and the earlier research reviewed in Chapter 2. The purpose is to demonstrate how the findings match available knowledge and in what ways the work makes some new contributions.

Previous studies on IoRT and edge AI, such as Atzori et al. (2021), concentrated largely on the high-level architecture and integration issues. The results of this thesis support those findings by showing that real-time IoRT systems need efficient communication between modules and careful use of hardware resources. The modular design of this work is based on the suggestions of these studies and shows that these designs are possible to apply to practical edge hardware.

Studies on lightweight LLMs, such as TinyBERT and DistilBERT, have shown that small transformer models can achieve good accuracy with limited resources. The results of this thesis confirm this claim, because DistilBERT performed reliably on Raspberry Pi 5 and understood short user commands with low latency. This aligns with the findings of Jiao et al. (2020), who showed that distilled models are suitable for embedded applications.

Research on computer vision models for edge devices, such as MobileNet and Tiny-YOLO, highlighted the importance of balancing speed and accuracy. The observations are in line with the experiments performed in this thesis. Even without the use of the GPU, YOLOv8n was more than faster and delivered acceptable results in terms of accuracy. This is in line with previous findings published by Howard et al. (2017) and Ultralytics (2023), who observed that lightweight models may still be used to achieve great performance with regard to real-time object detection.

The findings on predictive analytics also support previous research. Random Forest has been very popular in making predictions and detecting anomalies on small datasets due to its simplicity and speed. The results of this thesis confirm that Random Forest is a good choice for IoRT environments where sensor data arrive in small windows and must be processed quickly.

The difference between this thesis and previous studies lies in the fact that all three elements of AI combining LLM, CV, and predictive ML are implemented in practice, and a single multimodal system is developed to work on an edge device. Most of the previous research concentrated either on

single-modal performance or on cloud-based systems. This work offers effective evidence, by demonstrating that multimodal integration can be done on Raspberry Pi 5 which proves that edge devices can enable more intelligent IoRT behavior. This input assists in bridging the gap between theory and practice.

7 Conclusion and Future Work

This chapter summarizes the results of the study, highlights the main contributions, and discusses the limitations of the work. It also presents suggestions for future research that could extend the multimodal IoRT framework developed in this thesis. The idea is to look back at what has been accomplished, what still has to be done, and how the system can be improved in the future.

7.1 Summary of Findings

This thesis studied how lightweight artificial intelligence models can work together in a small Internet of Robotic Things system. The primary objective was to implement a language model, a computer vision model and a predictive machine learning model on Raspberry Pi 5 and to check whether the system can make decisions in real-time. The results showed that the multimodal system worked well even though the device had limited resources.

The system was able to understand short user commands, detect objects in the camera view and check sensor values at the same time. The tests revealed that every module performed their tasks properly. The full system also reacted correctly to commands and produced safe actions. The decisions were created within a short time window, which is important for IoRT tasks that need fast responses.

The test scenarios demonstrated the behavior of the system in various circumstances. The system completed the task when the object was present. The system waited and took no action when the object was missing. The system prevented the act when the sensors indicated unsafe values. In cases where the inputs were noisy or even late, the system remained stable. The fact that these behaviors demonstrate that the combination of text, camera data, and sensor values assists the system in making decisions that are better and safer. The Raspberry Pi 5 was capable of handling all the

modules without a cooling fan, meaning that lightweight multimodal AI can be implemented on inexpensive hardware.

7.2 Contributions to Research and Practice

This thesis provides several useful contributions. First, it provides a practical demonstration of the way lightweight multimodal AI can be deployed on an edge device. Cloud-based models are commonly addressed in IoRT studies, yet this paper demonstrates that small models can also be used in real time. It is helpful in those cases when organizations cannot always rely on cloud services. The findings also confirm that edge computing may decrease latency and enhance safety.

Second, the system design is modular. Each of the modules works independently and communicates via MQTT. This implies that, any section of the system can be substituted in the future without altering the entire system. It is useful in long-term projects that require updating. This work also demonstrates the collaboration of language and vision and sensor models. This combination can help solve more complex IoRT tasks that need better understanding of the environment.

Third, the thesis shows a development process that other students or engineers can follow. The work involves the selection of models, integration processes, message forms, fusion logic and testing procedures. These measures would assist others in developing similar systems to educate, research or small industry work. The results also show that multimodal systems can improve safety and reduce incorrect actions, which is important in many workplaces.

7.3 Limitations

Every research and prototype system has its own limitations, and understanding these limits is important for improving the work in the future. Although the system worked well, there are some limitations. The experiments were tested on a Raspberry Pi 5 that did not have a cooling fan. The device got warm in long tests and this resulted in minor drops in speed. To increase stability a cooling fan, or even a small heat sink would help. The models used in this thesis were lightweight versions. Stronger models could give better accuracy, but they might not run well on a small device.

The datasets used for testing were small and some were synthetic. More accurate results would be achieved by using bigger datasets of the real environment. The fusion node also used simple rule-based logic. This was effective with the test cases, but more complicated tasks may require a more advanced fusion procedure. The system was experimented in simple indoor environment. The actual factories or the natural world might generate more noises and unforeseen information.

Another limitation is that the system was tested with only one camera and one sensor type at a time. A real IoRT system can be equipped with many sensors simultaneously. This would require powerful fusion technique and more experimentation. The results of this thesis are good for small prototypes, but they do not show full performance in large scale industry settings.

7.4 Future Research Directions

Based on the results and limitations of this study, several directions can be explored to improve the system and expand its capabilities. This work can be continued in many ways. One of them is to test the system with more powerful devices, like NVIDIA Jetson boards or Coral TPUs. These devices have better support for AI models and could increase accuracy and speed. Another idea is to add more sensors, such as thermal sensors, pressure sensors or depth cameras. This would make the system more useful for different types of IoRT tasks.

The fusion node can also be improved. A fusion approach that relies on machine learning might be capable of learning more effective patterns between text, sight and sensors. This can enhance the quality of decision in more complicated tasks. Speech input, gesture recognition or tracking could also be added to a future system. These features would make the system better for human robot interaction.

The other way is to experiment the system in actual industrial settings. It may involve intelligent warehouses, mini robots or automated inspection conveyor belts. Real world data would assist in validating the system and illustrating how it works outside the lab. Hybrid edge cloud configuration is also possible by connecting the system to cloud services when necessary. This would assist in balancing speed and accuracy.

Finally, this work can support future research on multimodal learning on small devices. The models are becoming smaller and efficient. Multimodal IoRT systems may become more common in industry. The results from this thesis can help guide these developments and show how simple systems can still produce good results.

Overall, this research adds useful insights to the related field of IoRT by demonstrating that multimodal AI can be directly executed on edge devices. The system developed in this thesis can serve as a starting point for future improvements and more advanced multimodal IoRT applications.

References

Atzori, L., Iera, A., & Morabito, G. (2021). The Internet of Things: A survey. *Computer Networks, 148*, 133–146. https://doi.org/10.1016/j.comnet.2021.01.001

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901. https://arxiv.org/abs/2005.14165

Chen, X., & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE, 107*(8), 1655–1674. https://doi.org/10.1109/JPROC.2019.2921977

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding* (arXiv:1810.04805). https://arxiv.org/abs/1810.04805

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2020). Adversarial examples against object detection: A survey. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2020.2970675

European Commission. (2021). *Ethics guidelines for trustworthy AI*. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

Finnish National Board on Research Integrity (TENK). (2019). *The ethical principles of research with human participants and ethical review in the human sciences in Finland*. https://tenk.fi/en/ethical-review

García, L., Alonso, R. S., Prieto, J., & Corchado, J. M. (2020). A survey on the applications of the Internet of Robotic Things. *Applied Sciences, 10*(16), Article 5668. https://doi.org/10.3390/app10165668

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. https://www.deeplearningbook.org/

Han, S., Mao, H., & Dally, W. J. (2016). *Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding* (arXiv:1510.00149). https://arxiv.org/abs/1510.00149

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). *MobileNets: Efficient convolutional neural networks for mobile vision applications* (arXiv:1704.04861). https://arxiv.org/abs/1704.04861

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters* (arXiv:1602.07360). https://arxiv.org/abs/1602.07360

JAMK University of Applied Sciences. (2023). *Responsible conduct of research and data management guidelines*.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., … Liu, Q. (2020). *TinyBERT: Distilling BERT for natural language understanding* (arXiv:1909.10351). https://arxiv.org/abs/1909.10351

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lin, T. Y., Maire, M., Belongie, J., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)* (pp. 740–755). https://doi.org/10.1007/978-3-319-10602-1_48

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single-shot multibox detector. In *European Conference on Computer Vision (ECCV)* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

Oates, B. J. (2006). *Researching information systems and computing*. SAGE Publications.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., … Ng, A. Y. (2009). ROS: An open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91

Robson, C., & McCartan, K. (2016). *Real world research* (4th ed.). Wiley.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT: A distilled version of BERT: Smaller, faster, cheaper and lighter* (arXiv:1910.01108). https://arxiv.org/abs/1910.01108

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer, 50*(1), 30–39. https://doi.org/10.1109/MC.2017.9

Sharshar, A., Khan, L. U., Ullah, W., & Guizani, M. (2025). *Vision-language models for edge networks: A comprehensive survey* (arXiv:2502.07855). https://arxiv.org/abs/2502.07855

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., Zhou, D., … Zhou, J. (2020). *MobileBERT: A compact task-agnostic BERT for resource-limited devices* (arXiv:2004.02984). https://arxiv.org/abs/2004.02984

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9). https://doi.org/10.1109/CVPR.2015.7298594

Ultralytics. (2023). *YOLOv8 documentation*. https://docs.ultralytics.com/

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2020). *YOLOv4: Optimal speed and accuracy of object detection* (arXiv:2004.10934). https://arxiv.org/abs/2004.10934

Zhou, Z., Chen, X., & Li, E. (2020). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE, 107*(8), 1738–1762. https://doi.org/10.1109/JPROC.2019.2951977

Appendices

The appendices are used to present supplementary information which includes small code samples, sample data points, model setup, and detailed figures. These are included to enable the reader to understand the implementation, but they are separated by the main chapters to ensure readability.

Appendix 1. Code Snapshots

This appendix shows some of the parts of the code that are used in the multimodal IoRT system. The most significant parts are considered, including model loading, inference loops, MQTT communication, and fusion logic. The complete source code is not provided since it is too large, but these examples demonstrate the major steps that are required to replicate the system.

Model Loading (LLM Example)

```
from transformers import DistilBertForSequenceClassification,
DistilBertTokenizer
import torch

tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")
model =
DistilBertForSequenceClassification.from_pretrained("./models/llm_intent")
model.eval()

def predict_intent(text):
    inputs = tokenizer(text, return_tensors="pt")
    with torch.no_grad():
        outputs = model(**inputs)
    label_id = outputs.logits.argmax().item()
    return label_id
```

Vision Inference (YOLOv8n Example)

```
from ultralytics import YOLO
import cv2

model = YOLO("./models/yolov8n.pt")

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    results = model(frame)
    print(results)
```

Predictive Model (Random Forest)

```
import joblib
import numpy as np

rf_model = joblib.load("./models/rf_anomaly.pkl")

def check_sensor(values):
    arr = np.array(values).reshape(1, -1)
    prediction = rf_model.predict(arr)
    return prediction[0]
```

MQTT Message Publish/Subscribe

```
import paho.mqtt.client as mqtt
import json
import time

client = mqtt.Client()
client.connect("localhost", 1883)

def publish_result(topic, data):
    payload = json.dumps(data)
    client.publish(topic, payload)
```

Fusion Logic (Simplified)

```
def fuse_outputs(llm, vision, sensor):
    if sensor["status"] == "anomaly":
        return "halt_operation"
    if llm["intent"] == "inspect_tool" and vision["object"] == "tool":
        return "move_to_tool"
    return "neutral"
```

Appendix 2. Dataset Samples

This appendix includes small samples of the datasets that were used in this thesis. The samples consist of fragments of sensor logs, sample names of image files, and intent test command sentences. Representative examples are presented only, as entire datasets are too big and unnecessary to the reader.

Sample Sensor Log (CSV)

```
timestamp,vibration
2025-11-10T14:20:10.215,0.02
2025-11-10T14:20:10.316,0.03
2025-11-10T14:20:10.417,0.10   <- anomaly spike
2025-11-10T14:20:10.518,0.02
```

Sample Vision Images (Paths)

```
datasets/images/tool_01.jpg
datasets/images/tool_02.jpg
datasets/images/cup.jpg
```

Sample Command Text for LLM

```
inspect tool
check object
start process
stop process
scan area
```

Appendix 3. Model Configurations

This appendix contains the key configuration parameters used in the three artificial intelligence models of the system. They are DistilBERT (intent recognition), YOLOv8n (object detection), and Random Forest (anomaly detection). Fusion rules and thresholds are also provided to enable readers to know how the system makes decision.

DistilBERT Configuration

- Batch size: 16
- Max sequence length: 128
- Learning rate: 5e-5
- Training epochs: 10
- Task: Intent classification

YOLOv8n (Vision Model)

- Image size: 640 x 640
- Confidence threshold: 0.25
- IoU threshold: 0.45
- Size: around 3.3M parameters

Random Forest (Predictive Model)

- Number of trees: 100
- Max depth: None
- Criterion: Gini
- Input features: Sensor values

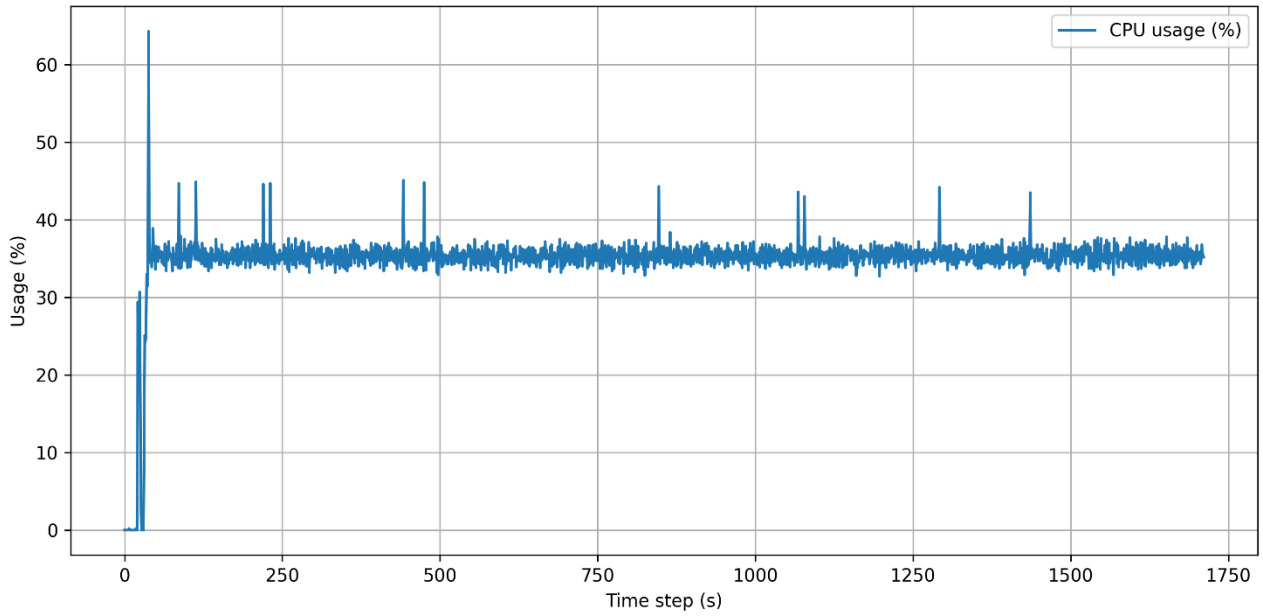
Fusion Settings

- Time window: 300 ms
- LLM minimum confidence: 0.65
- Vision minimum confidence: 0.40
- Anomaly threshold: Based on RF classifier output

Appendix 4. Additional Figures or Tables

This appendix provides extra figures and logs that support the results presented in Chapter 5. These are full-size CPU and RAM usage graphs, and MQTT message monitoring screenshots. These materials are more detailed to readers who are interested in the performance of the system.

CPU Usage (Full Graph)



RAM Usage (Full Graph)