## Plan Overview

*A Data Management Plan created using DMPTuuli*

**Title:** Edge AI for Smart IoT Monitoring Systems: Evaluating Lightweight AI Models

**Creator:**md zia uddin muzumder

**Affiliation:** Metropolia University of Applied Sciences

**Template:** DMP for Thesis

**Project abstract:**

This thesis examined lightweight AI models for Edge AI-based smart IoT monitoring systems. The study was motivated by the limitations of cloud-based monitoring, especially in applications requiring low latency, reduced bandwidth use, privacy-aware processing, and reliable operation under constrained connectivity. Since edge devices are resource-constrained, selecting suitable lightweight AI models is an important practical challenge.

The aim of the thesis was to investigate lightweight AI models for real-time monitoring and decision making on edge devices. The study used a structured literature review supported by a comparative analytical framework. Relevant studies were selected and analyzed based on criteria such as latency, memory footprint, energy suitability, task performance, and deployment practicality.

The results show that no single lightweight AI model is universally best for all smart IoT monitoring applications. Instead, model suitability depends on the monitoring task, hardware capability, latency requirements, memory limits, energy constraints, and deployment context. Based on the findings, the thesis developed an Edge AI model comparison table, an Edge AI Suitability score, an Edge AI Decision Flow Model, a research gap analysis, and a proposed smart monitoring architecture.The thesis concludes that lightweight Edge AI should be approached through structured, deployment-aware model selection rather than through accuracy-only comparison.

**ID:** 31577

**Last modified:** 18-05-2026

**Copyright information:**

# Edge AI for Smart IoT Monitoring Systems: Evaluating Lightweight AI Models

## 1. General description of the data

### 1.1. Briefly describe the data you collect and/or produce or that already exists, as well as their properties (type, file format, size, access rights, collection methods). Create a table or list of the data.

**Appendices**

**Appendix 1. Research Design of the Thesis**

This thesis follows a structured literature review design supported by a comparative analytical framework. The purpose of the research design is to investigate lightweight Edge AI models for smart IoT monitoring systems and to compare their suitability for real-time monitoring and decision making on resource-constrained edge devices.

The research design includes the following stages:

| Stage | Description | Output |
|---|---|---|
| 1. Topic framing | Define the research problem, aim, scope, and research questions | Research focus and thesis direction |
| 2. Literature search | Search academic databases using predefined search terms | Initial list of relevant papers |
| 3. Screening and selection | Apply inclusion and exclusion criteria | Final set of selected studies |
| 4. Data extraction | Extract information on model type, hardware, latency, memory, energy, and accuracy | Comparison matrix |
| 5. Comparative analysis | Compare selected studies using common evaluation dimensions | Edge AI model comparison table |
| 6. Synthesis | Develop practical analytical outputs from the reviewed literature | EAIS, decision flow model, research gaps, and proposed architecture |

This design supports the aim of the thesis by making the review process transparent and by turning scattered literature findings into structured guidance for Edge AI model selection.

**Appendix 2. Literature Review Process Flow**

The literature review process followed a structured sequence from topic definition to final synthesis.

Define thesis focus
↓
Define search terms
↓
Search academic databases
↓
Screen titles
↓
Screen abstracts
↓
Review full texts
↓
Apply inclusion and exclusion criteria
↓
Select final studies
↓
Extract comparison data
↓
Compare models and optimization methods
↓
Synthesize findings into thesis outputs

The purpose of this process was to make the literature review transparent, consistent, and easier to justify. The final synthesis supported the development of the Edge AI model comparison table, the Edge AI Suitability score, the decision flow model, the research gap analysis, and the proposed smart monitoring architecture.

**Appendix 3. Preliminary Search Strings**

| No. | Search String |
|---|---|
| 1 | “Edge AI” AND “IoT monitoring” |
| 2 | “Edge artificial intelligence” AND “smart monitoring systems” |
| 3 | “TinyML” AND “real-time monitoring” |
| 4 | “Tiny machine learning” AND “IoT devices” |
| 5 | “Lightweight neural network” AND “edge device” |
| 6 | “Lightweight deep learning” AND “IoT monitoring” |
| 7 | “On-device inference” AND “Internet of Things” |
| 8 | “Embedded AI” AND “resource-constrained devices” |
| 9 | “Resource-constrained machine learning” AND “monitoring system” |
| 10 | “Edge intelligence” AND “smart IoT” |
| 11 | “Model compression” AND “Edge AI” |
| 12 | “Quantization” AND “edge deployment” |
| 13 | “Pruning” AND “lightweight neural networks” |
| 14 | “Knowledge distillation” AND “edge inference” |
| 15 | “Hardware-aware neural architecture search” AND “edge devices” |
| 16 | “MobileNet” AND “edge inference” |
| 17 | “MobileNetV2” AND “IoT monitoring” |
| 18 | “MCUNet” AND “TinyML” |
| 19 | “EfficientNet” AND “edge AI” |
| 20 | “SqueezeNet” AND “embedded inference” |

The search strings were used in academic databases such as IEEE Xplore, ScienceDirect, SpringerLink, ACM Digital Library, and Google Scholar. The strings were adjusted slightly depending on the search interface of each database.

**Appendix 4. Inclusion and Exclusion Criteria Table**

| Category | Inclusion Criteria | Exclusion Criteria |
|---|---|---|
| Topic relevance | Studies related to Edge AI, TinyML, lightweight AI models, embedded AI, or on-device inference | Studies focused only on general AI or cloud AI without edge deployment relevance |
| Application relevance | Studies related to IoT monitoring, smart monitoring, real-time inference, anomaly detection, event recognition, or decision making | Studies unrelated to IoT monitoring or smart sensing applications |
| Deployment relevance | Studies discussing resource-constrained devices, microcontrollers, embedded boards, gateways, or edge hardware | Studies focused only on high-performance cloud servers or data centers |
| Model relevance | Studies involving lightweight models such as MobileNet, SqueezeNet, EfficientNet-style models, MCUNet, TinyML models, or compressed neural networks | Studies using only large deep learning models without optimization or edge suitability discussion |
| Optimization relevance | Studies discussing quantization, pruning, knowledge distillation, hardware-aware NAS, or model compression | Studies with no technical or deployment-related analysis |
| Evaluation relevance | Studies reporting or discussing latency, memory footprint, model size, energy efficiency, accuracy, computational cost, or deployment practicality | Studies that do not provide enough information for comparison |
| Publication type | Peer-reviewed journal articles, conference papers, survey papers, and technically reliable sources | Duplicates, inaccessible papers, non-technical blog posts, or clearly unreliable sources |
| Language | English-language publications | Non-English publications not accessible for analysis |

**Appendix 5. Data Extraction and Comparison Matrix**

| Study | Application Domain | Monitoring Task | Model / Approach | Optimization Method | Hardware Platform | Performance Metric | Latency | Memory / Model Size | Energy Suitability | Main Strength | Main Limitation | Relevance to Thesis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Howard et al., 2017 | General mobile / edge vision | Image classification | MobileNet | Efficient architecture design | Mobile / embedded-class devices | Accuracy and computational cost | Low to medium | Low | Good | Strong balance between efficiency and performance | May still be heavy for very small MCUs | Supports compact CNN comparison |
| Sandler et al., 2018 | Mobile / embedded vision | Image classification and detection | MobileNetV2 | Inverted residuals and linear bottlenecks | Mobile / embedded devices | Accuracy and efficiency | Low to medium | Low | Good | Efficient architecture for constrained vision tasks | Not ideal for ultra-constrained MCUs | Supports lightweight model evaluation |
| Iandola et al., 2016 | Compact vision models | Image classification | SqueezeNet | Compact CNN design | Embedded / low-memory devices | Accuracy and model size | Low | Very low | Good | Very small parameter size | Lower performance than newer compact models | Supports low-memory model comparison |
| Tan and Le, 2019 | Efficient vision models | Image classification | EfficientNet | Compound scaling | Stronger embedded / edge devices | Accuracy and efficiency | Medium | Medium | Moderate | Strong accuracy-efficiency balance | Heavier than TinyML models | Supports efficient scaling discussion |
| Lin et al., 2020 | TinyML / embedded AI | Tiny visual and sensor inference | MCUNet | Architecture-runtime co-design | Microcontrollers | Accuracy and deployment feasibility | Very low | Very low | Very high | Designed for constrained devices | Limited task complexity | Supports ultra-constrained Edge AI analysis |
| Heydari and Mahmoud, 2025 | TinyML / on-device inference | Survey of TinyML applications | TinyML approaches | Quantization, compression, deployment optimization | MCUs and edge devices | Survey-based comparison | Varies | Varies | High relevance | Broad overview of TinyML deployment | Survey-level, not one benchmark | Supports gaps and future work |
| Merenda et al., 2020 | Edge ML / AIoT | Edge inference and IoT applications | Edge machine learning | Model optimization and edge deployment | Edge devices and IoT nodes | Survey-based comparison | Varies | Varies | High relevance | Strong overview of Edge AI motivations | Broad survey scope | Supports theoretical background |

**Appendix 6. Simplified IoT Monitoring System Pipeline**

This appendix presents a simplified pipeline for a typical IoT monitoring system.

| Stage | Function | Example Components |
|---|---|---|
| 1. Sensing | Collects data from the physical environment | Temperature sensors, cameras, microphones, motion sensors, industrial sensors |
| 2. Transmission | Sends collected data through communication networks | Wi-Fi, Bluetooth Low Energy, LoRaWAN, Zigbee, 5G |
| 3. Aggregation and Processing | Stores, filters, and processes data | Gateways, cloud servers, edge nodes, databases |
| 4. Analytical Output | Produces useful outcomes for users or systems | Alerts, dashboards, predictions, classifications, control actions |

Physical Environment

↓

Sensors and IoT Devices

↓

Network Transmission

↓

Gateway / Cloud / Edge Processing

↓

Dashboard, Alert, Prediction, or Decision

This pipeline shows how data usually moves from physical sensing to useful monitoring output.

**Appendix 7. Key Application Domains of IoT Monitoring**

| Domain | Example Use Cases | Typical Monitoring Tasks |
|---|---|---|
| Industrial monitoring | Predictive maintenance, machine health tracking, process monitoring | Anomaly detection, fault prediction, equipment status classification |
| Environmental monitoring | Air quality, water quality, weather observation, wildfire detection | Sensor data analysis, threshold detection, trend monitoring |
| Smart cities | Traffic monitoring, waste management, energy distribution, infrastructure observation | Event detection, flow analysis, resource optimization |
| Healthcare and wellbeing | Remote patient monitoring, wearable sensing, activity tracking | Physiological signal monitoring, alert generation, pattern detection |
| Agriculture | Soil monitoring, irrigation control, crop condition tracking | Environmental sensing, disease detection, resource control |

**Appendix 8. Key Limitations of Traditional Cloud-Based IoT Monitoring**

| Limitation | Explanation | Practical Impact |
|---|---|---|
| Latency | Data must travel to the cloud and back before action can be taken | Delayed response in time-sensitive monitoring |
| Bandwidth consumption | Raw sensor, image, or video data may require continuous transmission | Increased network load and cost |
| Connectivity dependence | Cloud-based systems need stable internet access | Reduced reliability in remote or unstable environments |
| Privacy concerns | Sensitive data may leave the local environment | Higher risk of exposure or compliance issues |
| Scalability overhead | Large device networks generate large amounts of data | More storage, processing, and transmission burden |
| Energy inefficiency | Frequent data transmission consumes power | Shorter battery life in low-power devices |

**Appendix 9. Shift from Cloud-Centric Monitoring to Edge-Supported Smart Monitoring**

| Feature | Cloud-Centric Monitoring | Edge-Supported Smart Monitoring |
|---|---|---|
| Main processing location | Cloud server | Device, gateway, and cloud |
| Raw data transfer | Usually high | Reduced through local filtering |
| Latency | Higher due to round-trip communication | Lower due to local inference |
| Connectivity dependence | High | Lower |
| Privacy | Raw data often leaves local environment | More data can stay local |
| Scalability | Cloud resources scale, but transmission load increases | Distributed processing reduces upstream load |
| Main role of cloud | Storage, analytics, decision-making | Training, long-term storage, coordination, dashboards |
| Main role of edge | Usually limited | Local inference, filtering, alerting, immediate decisions |

Cloud-Centric Model:
Sensor → Network → Cloud Processing → Result → Action

Edge-Supported Model:
Sensor → Local Edge Inference → Gateway Coordination → Cloud Analytics when needed

**Appendix 10. Cloud–Edge–Endpoint Intelligence Overview**

| Layer | Main Role | Example Components | Typical AI Function |
|---|---|---|---|
| Endpoint | Collects raw data and may perform simple local processing | Sensors, microcontrollers, cameras, wearables | Threshold detection, tiny inference |
| Edge node | Performs local inference and filtering near the data source | Raspberry Pi, Jetson Nano, embedded gateway | Classification, anomaly detection, event recognition |
| Cloud | Performs large-scale storage, training, and system-wide analytics | Cloud servers, data lakes, dashboards | Model training, historical analysis, central coordination |

Endpoint Device
↓ local data collection
Edge Node / Gateway
↓ filtered results or alerts
Cloud Platform
↓ model updates, dashboards, long-term analytics
System-level Decision Support

**Appendix 11. Simplified Smart IoT Monitoring Architecture**

| Layer | Description | Example |
|---|---|---|
| Sensing layer | Collects data from the physical environment | Temperature sensor, sound sensor, camera |
| Communication layer | Transfers data between devices and processing components | Wi-Fi, Bluetooth, LoRaWAN, MQTT |
| Processing layer | Filters, analyzes, or classifies data | Edge inference, gateway processing, cloud analytics |
| Application layer | Presents results or triggers actions | Dashboard, alert system, automated control |

Sensing Layer
↓
Communication Layer
↓
Processing Layer
↓
Application Layer

This architecture shows the basic structure of smart IoT monitoring systems and where Edge AI can be integrated.

**Appendix 12. Key Lightweight AI Model Families**

| Model / Family | Main Design Idea | Typical Strength | Main Limitation | Suitable Edge Context |
|---|---|---|---|---|
| MobileNet | Depthwise separable convolutions | Good balance between accuracy and efficiency | May still be heavy for very small MCUs | Mobile and embedded vision |
| MobileNetV2 | Inverted residuals and linear bottlenecks | Efficient compact CNN performance | Not ideal for ultra-low-memory MCUs | Edge image classification |
| MobileNetV3 | Hardware-aware architecture refinement | Improved mobile efficiency | More complex than earlier versions | Mobile and stronger edge devices |
| SqueezeNet | Very small parameter count | Low memory footprint | Lower performance than newer models | Low-memory embedded devices |
| EfficientNet | Compound scaling of depth, width, and resolution | Strong accuracy-efficiency balance | Standard variants may be too heavy for constrained devices | Stronger edge boards |
| MCUNet | Model-runtime co-design for microcontrollers | Very suitable for ultra-constrained hardware | Limited task complexity | TinyML and MCU deployment |
| TinyML models | Small models for tiny devices | Very low memory and energy use | Limited flexibility | Always-on sensing and microcontrollers |

**Appendix 13. Optimization Techniques for Edge Deployment**

| Technique | Description | Benefit | Limitation |
|---|---|---|---|
| Post-training quantization | Reduces precision after model training | Easy to apply, reduces size and inference cost | May reduce accuracy |
| Quantization-aware training | Simulates quantization during training | Better accuracy retention than PTQ | Requires training access and more effort |
| Structured pruning | Removes channels, filters, or layers | More practical speedup on standard hardware | Requires careful tuning |
| Unstructured pruning | Removes individual weights | Can reduce parameter count greatly | May not speed up inference without sparse hardware support |
| Knowledge distillation | Trains a small model using a larger teacher model | Helps preserve performance in smaller models | Requires teacher model and training process |
| Hardware-aware NAS | Searches for architectures suited to target hardware | Better hardware-specific efficiency | Computationally complex |
| Once-for-All | Trains one supernetwork and specializes subnetworks | Flexible deployment across devices | Requires complex training setup |
| ProxylessNAS | Searches architectures directly for target task and hardware | Device-aware model design | More complex than manual model selection |

**Appendix 14. Proposed Edge Intelligence Levels**

| Level | Name | Description | Example Monitoring Task |
|---|---|---|---|
| Level 1 | Basic reactive local processing | Rule-based or threshold-based local action without learned inference | Temperature threshold alert |
| Level 2 | Compact learned inference | Small ML models perform simple classification or anomaly detection | Binary occupancy detection |
| Level 3 | Lightweight deep learning at the edge | Compact CNNs or TinyML models perform richer feature-based inference | Visual anomaly detection, keyword spotting |
| Level 4 | Adaptive and collaborative edge intelligence | Edge and cloud components coordinate, adapt, or share intelligence | Multi-device monitoring with adaptive model updates |

This framework is proposed as a thesis-specific conceptual tool, not as a standardized taxonomy.

**Appendix 15. Summary of Analyzed Paper Groups**

| Paper Group | Purpose in Thesis | Example Sources |
|---|---|---|
| Edge AI / AIoT survey papers | Explain motivations, architecture, and general challenges | Merenda et al., 2020; Marengo et al., 2024 |
| TinyML and on-device AI surveys | Explain microcontroller and ultra-constrained deployment | Heydari and Mahmoud, 2025 |
| Lightweight model papers | Provide model-family evidence | Howard et al., 2017; Sandler et al., 2018; Iandola et al., 2016; Tan and Le, 2019; Lin et al., 2020 |
| Optimization method papers | Support compression and hardware-aware deployment discussion | Hinton et al., 2015; Cai et al., 2018; Cai et al., 2020 |
| Application-focused studies | Provide context for monitoring scenarios | IoT monitoring and smart sensing papers |

**Appendix 16. Extended Edge AI Comparison Matrix**

| Model / Approach | Typical Task | Performance | Latency | Memory | Energy | Best-Fit Device | Best Use Case |
|---|---|---|---|---|---|---|---|
| MobileNet | Image classification, event detection | High | Low-medium | Low | Good | Raspberry Pi, mobile-class devices | General embedded vision |
| MobileNetV2 | Visual monitoring, anomaly detection | High | Low-medium | Low | Good | Raspberry Pi, Jetson Nano | Efficient image-based monitoring |
| SqueezeNet | Simple visual classification | Medium-high | Low | Very low | Good | Low-memory embedded devices | Compact visual inference |
| EfficientNet-style compact model | Advanced image classification | High | Medium | Medium | Moderate | Jetson-class devices | Stronger compact edge vision |
| MCUNet | Tiny vision and sensor inference | Medium | Very low | Very low | Very high | Microcontrollers | Ultra-constrained sensing |
| Quantized CNN | Always-on monitoring | Medium-high | Very low | Very low | Very high | MCU / low-power accelerators | Low-power real-time inference |
| Pruned model | Mixed monitoring tasks | Medium-high | Low | Low | Good | Mid-range embedded devices | Deployment-optimized inference |
| Distilled model | Mixed monitoring tasks | Medium-high | Low | Low | Good | Embedded edge devices | Smaller student model deployment |

**Appendix 17. Edge AI Suitability Score Template**

The Edge AI Suitability score uses five dimensions:

| Symbol | Dimension | Meaning |
|---|---|---|
| P | Task Performance | Accuracy, F1-score, detection rate, or task-specific performance |
| L | Latency Suitability | Suitability for real-time or near-real-time inference |
| M | Memory Suitability | Compatibility with limited device memory |
| E | Energy Suitability | Suitability for low-power or battery-based operation |
| D | Deployment Practicality | Ease of deployment based on toolchains, hardware support, and integration |

Formula:

EAIS = (P + L + M + E + D) / 5

| Score Range | Interpretation |
|---|---|
| 4.5–5.0 | Excellent edge suitability |
| 3.5–4.4 | Strong practical suitability |
| 2.5–3.4 | Moderate suitability, context-dependent |
| 1.5–2.4 | Weak suitability for constrained edge deployment |
| 1.0–1.4 | Poor edge suitability |

Blank scoring template:

| Model / Approach | P | L | M | E | D | EAIS |
|---|---|---|---|---|---|---|
| Model 1 | | | | | | |
| Model 2 | | | | | | |
| Model 3 | | | | | | |

**Appendix 18. Edge AI Decision Flow Model**

Step 1: Identify the monitoring task
↓
Step 2: Identify the hardware class
↓
Step 3: Identify the dominant deployment constraint
↓
Step 4: Select the initial model family
↓
Step 5: Apply deployment optimization
↓
Step 6: Validate suitability using EAIS

Expanded decision logic:

| Step | Decision Question | Example Outcome |
|---|---|---|
| 1 | What type of monitoring task is required? | Image, audio, sensor, time-series, event detection |
| 2 | What hardware class is available? | MCU, Raspberry Pi, Jetson, gateway |
| 3 | What is the main constraint? | Latency, memory, energy, accuracy, deployment practicality |
| 4 | Which model family fits first? | TinyML, MCUNet, MobileNet, SqueezeNet, EfficientNet-style model |
| 5 | What optimization is needed? | Quantization, pruning, distillation |
| 6 | Does the model fit the target scenario? | Evaluate using EAIS |

**Appendix 19. Conceptual Energy vs Accuracy Trade-off Curve**

This appendix describes the conceptual relationship between energy efficiency and predictive performance in lightweight Edge AI deployment.

```
High Accuracy
  |
  |                Larger compact models
  |             EfficientNet-style approaches
  |
  |         MobileNet / MobileNetV2
  |
  |     SqueezeNet / Quantized CNN
  |
  | MCUNet / TinyML
  |
  |_______________________________________
    Very low energy        Higher energy
```

The curve is conceptual and is not based on direct empirical measurements carried out in this thesis. It is based on patterns observed in the reviewed literature. The main purpose is to illustrate that higher predictive performance often requires more computation, memory, and energy, while ultra-low-power models may sacrifice some predictive richness.

**Appendix 20. Research Gaps and Practical Challenges Summary Table**

| Gap / Challenge | Description | Source Type |
|---|---|---|
| Lack of standardized comparison criteria | Studies use different datasets, devices, metrics, and reporting formats | Literature-reported and author-synthesized |
| Limited real-world validation | Many studies rely on benchmark settings rather than long-term deployment | Author-synthesized |
| Incomplete energy reporting | Energy efficiency is often discussed but not consistently measured | Literature-reported and author-synthesized |
| Hardware fragmentation | Results vary across hardware platforms, accelerators, and runtimes | Literature-reported and author-synthesized |
| Monitoring-task diversity | IoT monitoring includes image, audio, sensor, event, and time-series tasks | Author-synthesized |
| Limited practical selection guidance | Few studies turn findings into model-selection support for practitioners | Author-synthesized |

**Appendix 21. Proposed Edge AI Smart Monitoring Architecture**

| Layer | Function | Example Components | Role of AI |
|---|---|---|---|
| Sensing layer | Collects raw data | Sensors, cameras, microphones, wearables | Data generation |
| Edge inference layer | Performs local AI inference | MCU, Raspberry Pi, Jetson Nano, embedded device | Classification, anomaly detection, event recognition |
| Edge coordination / gateway layer | Aggregates and coordinates outputs | Local gateway, edge server, MQTT broker | Filtering, routing, local decision support |
| Cloud / central analytics layer | Performs long-term analysis and management | Cloud server, dashboard, database | Model training, historical analytics, system coordination |

Sensing Layer
↓
Edge Inference Layer
↓
Edge Coordination / Gateway Layer
↓
Cloud / Central Analytics Layer

This architecture supports a hybrid Edge AI monitoring approach where immediate inference happens locally and broader analytics remain in the cloud.

**Appendix 22. Research Question to Findings Mapping**

| Research Question | Main Finding | Related Chapter / Section | Thesis Output |
|---|---|---|---|
| RQ1: Why is Edge AI needed in smart IoT monitoring? | Edge AI reduces latency, communication overhead, privacy risks, and cloud dependency | Chapters 3, 4, 6 | Motivation analysis |
| RQ2: Which lightweight models and optimization techniques are relevant? | MobileNet, SqueezeNet, EfficientNet-style models, MCUNet, TinyML, quantization, pruning, distillation, and NAS are relevant | Chapter 4 | Theoretical synthesis |
| RQ3: How do approaches compare across deployment dimensions? | Suitability depends on latency, memory, energy, performance, and deployment practicality | Chapter 5 | Comparison table and EAIS |
| RQ4: What gaps and challenges remain? | Standardization, energy reporting, real-world validation, hardware fragmentation, and task diversity remain key gaps | Chapter 5 and 6 | Research gap analysis |
| RQ5: How can practical model selection be supported? | Selection should begin with task, hardware class, dominant constraint, and optimization need | Chapter 5 and 6 | Decision flow model and architecture |

**Appendix 23. Future Work Directions Summary**

| Future Work Direction | Gap or Limitation Addressed |
|---|---|
| Standardized benchmarking across hardware classes and task types | Lack of consistent comparison criteria |
| More consistent energy reporting | Incomplete energy measurement and reporting |
| Task-specific model-selection studies | Monitoring-task diversity |
| Real-world long-term deployment validation | Limited field evidence |
| Empirical validation of EAIS and decision flow model | Thesis-specific tools need validation |
| Hybrid edge-cloud intelligence allocation | Need for better system-level design and coordination |

**1.2. How do you ensure the consistency and accuracy of the data?**

1.2 Ensuring consistency and accuracy of the data

Consistency and accuracy will be ensured by using a structured literature review process. Search strings, inclusion and exclusion criteria, and comparison dimensions are documented clearly. The selected studies will be screened according to predefined criteria, and extracted information will be organized in a comparison matrix. When numerical or technical values are used, they will be taken from the original publications or clearly described as literature-based qualitative assessments. Citations will be used to show the source of factual claims. The thesis will also avoid presenting illustrative or conceptual tables as direct empirical measurements.

## 2. Personal data, ethical principles and legislation

### 2.1. Does the data contain any personal information? If yes, enter in the supplementary information field: the personal data to be collected, whether there are special categories of personal data.

- Ei

No. The thesis does not collect or process personal data. The study is based on published academic and technical literature. No interviews, surveys, recordings, personal identifiers, contact details, health data, employment data, or other personal information will be collected.

### 2.2. Who has the main responsibility for the processing of personal data, i.e. controllership? If you do not collect personal data, you can skip this question.

Not applicable, because no personal data will be collected or processed in this thesis.

### 2.3. What measures are required to ensure data protection in my thesis? If you do not collect personal data, you can skip this question.

Not applicable for personal data, because no personal data will be collected. General data protection will still be maintained by storing thesis files securely and avoiding the inclusion of confidential or copyrighted full-text material in shared documents. Only properly cited summaries, extracted comparison information, and bibliographic references will be used in the thesis.

### 2.4. Do you need an ethical review? To the additional information field: justification and possible implementation

An ethical review is not required because the thesis does not involve human participants, personal data, interviews, surveys, experiments on people, sensitive data, medical data, or intervention-based research. The study is based on existing published literature and does not create direct risk to individuals or organizations.

### 2.5. Are there other research ethical questions related to the data?

The main research ethical questions relate to correct citation, plagiarism avoidance, accurate representation of sources, and responsible use of AI tools. All sources used in the thesis will be cited properly. The thesis will not copy large sections of published work. Any AI-assisted language editing or drafting support will be reviewed by the author, and the author remains responsible for the accuracy, originality, and final content of the thesis.

### 2.6. How will you manage the rights to the data you use, produce and share?

**Describe here how you will agree on the rights of the research data collected, produced and (re)used in the thesis. Describe the procedures for transferring rights that apply to your data.**

The thesis uses published literature according to normal academic citation practices. Copyrighted articles will not be redistributed as part of the thesis. Only summaries, citations, extracted comparison information, and author-created tables or frameworks will be included. The analytical outputs produced in the thesis, such as comparison tables, EAIS, decision flow model, and proposed architecture, are created by the thesis author based on the reviewed literature. Rights to the thesis text and author-created analytical outputs remain with the thesis author, unless otherwise required by Metropolia's thesis publication rules.

## 3. Data description and documentation

### 3.1. How do you describe and document the data in an understandable way?

The data will be documented through search records, search strings, inclusion and exclusion criteria, and a data extraction matrix. The thesis appendices will include key documentation such as the research design, literature review process flow, preliminary search strings, inclusion and exclusion criteria, and data extraction matrix. This documentation makes the review process understandable and transparent. The final thesis will also include references to all cited sources in the reference list.

## 4. Storage and data security during the thesis process

### 4.1. Describe here where the data will be stored and how it will be backed up during the thesis process.

The thesis files, literature notes, extraction matrix, and draft documents will be stored on the author’s personal computer and backed up using cloud storage, such as OneDrive, Google Drive, or another secure storage service. Important files will be saved in organized folders with version names or dates to avoid confusion between drafts. Published research papers will be stored only for personal study and citation purposes.

### 4.2. Who has access to your data, what can those people do with the data, and how do you ensure the safe transfer of the data to your potential collaborators?

The thesis author will have full access to the working files. Supervisors may receive selected thesis drafts, tables, and appendices for review. No personal or confidential data will be shared. If files are shared with supervisors, they will be transferred using secure university-approved systems, email, Wihi, or controlled cloud links. Access rights will be limited to necessary persons only.

## 5. Data after the thesis is completed: preserving, destruction or possible further use and opening

### 5.1. Is the data or part of it preserved? If yes: describe which data or part of it.

- Kyllä

The final thesis document, reference list, selected appendices, comparison tables, and author-created analytical outputs may be preserved as part of the final thesis submission. The final thesis will be published according to Metropolia’s thesis publication process, normally through Theseus unless otherwise instructed.

### 5.2. Will the data be destroyed or part of it? If yes, describe which data or part of the data, and how and when the destruction will occur.

- Kyllä

Unnecessary draft files, duplicate notes, temporary working files, and downloaded copies of articles that are no longer needed may be deleted after the thesis is completed and accepted. If any copyrighted full-text papers were downloaded for personal study, they will not be redistributed and may be removed from personal storage when no longer needed.

### 5.3. Are you planning to further use or open your data or part of it? If yes, describe which data or part of it. Also describe whether the data will be opened for public use with a license, or whether it will be handed over to a specific party for further use, as well as the possible measures that opening the data or preparing for further use requires (for example, anonymisation, permissions from research subjects, agreements with collaborators).

- Kyllä

The final thesis and its appendices may be publicly available through the official thesis publication platform. The thesis does not include personal data, confidential organizational data, or sensitive research data. The author-created analytical outputs, such as the comparison table, EAIS template, decision flow model, and proposed architecture, may be reused or developed further in future academic or professional work, with proper reference to the thesis. No separate anonymisation is needed because no personal data is collected.

## 6. Responsibilities and resources

### 6.1. Who is responsible for data management and what kind of resources does data management require?

The thesis author is responsible for managing the data during the thesis process. This includes organizing literature, storing files securely, keeping backups, documenting the review process, citing sources correctly, and ensuring that no confidential or personal data is included in the final thesis. The required resources are minimal and include a computer, word-processing software, reference management or citation tools if used, spreadsheet software for comparison tables, university database access, and secure storage or backup space.