Adaptive TinyML Model Distribution Protocol for Dynamic Environments

Project Proposal

Visal Gimhan Mediwake Rajapakse

w1742117 / 2018418

Supervisor: Guhanathan Poravi

Date: 3^{rd} November 2021

Department: Computer Science

Keywords: TinyML, Networking Protocols, Model Distribution

Contents

List of Figures

List of Tables

Acronyms

CC Cloud Computing.

DL Deep Learning.

IoT Internet of Things.

MCU Microcontroller Unit.

ML Machine Learning.

RAT Radio Access Technology.

TinyML Tiny Machine Learning.

1 Introduction

This proposal initially aims to provide the reader with a brief introduction to the problem domain of Tiny Machine Learning (TinyML) and conduct an initial survey on the existing literature. The problems identified with the state-of-the-art is then leveraged to present justifications as to why model updating protocols are necessary for TinyML to broaden the IoT network. By doing this that author aims to concretize the research significance and research gap.

2 Problem Domain

Internet of Things (IoT) has seen explosive growth in recent years and will continue to grow. This growth leads to IoTs inevitable penetration in our day-to-day lives. To provide some context, Cisco (2020) expects the number of devices per capita to grow from 3.6 by 2023, up from 2.4 in 2018. Provided this growth, the paramount quantities of data generated is already overwhelming conventional Cloud Computing (CC). However, the advent of Edge computing has brought processing closer to data generators, aiming to address the growing concerns and issues of CC.

2.1 EdgeML

Machine Learning (ML) plays an increasingly important role in many areas in the digital era we live in today. With the increase in processing power and new heights in energy efficiency, bringing ML to the edge is becoming an increasingly viable option. Edge ML is a component of Edge computing where data is processed at the source to get predictions/decisions. Doing so will reduce the strain on cloud computers while preserving data privacy and security with the benefit of low energy usage.

As much research in Edge ML has been conducted; researchers are now focusing on extending Edge applications by attempting to merge ML with even more compact and energy-efficient devices. To be specific, they are focusing on the Microcontroller Unit (MCU) class of technology. A key differentiator between this paradigm and Edge ML is that conventional ML cannot be applied to such devices as they are incredibly resource-constrained. Hence why the domain is dubbed Tiny Machine Learning (TinyML).

2.2 TinyML

TinyML – a subcategory of Edge computing, is formally defined as the "connecting point between constrained IoT devices and ML/DL without any resource-rich OS."(Doyu et al., 2021). Here, "resource-rich OS" is defined as operating systems such as macOS, Windows, and Linux.

For instance, conventional Machine Learning relies on well-configured computers with ample resources to train, test, and execute conventional Machine Learning models. TinyML, on the other hand, aims to challenge the stereotype that Machine Learning applications require far greater resources for inferencing by providing heavily resource-constrained devices with cognitive capabilities. Although inconspicuous, TinyML applications can be found in our day-to-day devices. A well-known example of such would be the voice prompts in personal assistants in smartphones, namely, Alexa, Siri, and Google Assistant (Zhang et al., 2017).

2.3 Connected oriented TinyML applications

Today, In the subject of TinyML, established frameworks such as Tensorflow Lite Micro (Tensorflow, 2020) allow a broader unification of embedded ML within an otherwise vastly heterogeneous and segmented MCU industry. Embedded systems, as mentioned, are the perfect candidate for low cost, low power, privacy-preserving IoT applications. Given that, due to the novelty of the research area, popular TinyML frameworks and most of the existing solutions in the literature assume that embedded systems permit only the inference of Machine learning and Deep learning algorithms (Disabato and Roveri, 2020; Ren et al., 2021).

Challenging the above assumption, research in TinyML is taking a different turn as solutions are now focusing on connection-oriented ML/DL (E.g, Collaborative learning and Task offloading) and lifelong learning capabilities, i.e., learning on the fly. An attempt to merge both Lifelong learning and Collaborative learning has been conducted by Kopparapu and Lin (2021) and Grau et al. (2021). In such solutions, communication with external nodes plays a pivotal role. However, despite the many research in energy-efficient communication protocols/schemes for IoT systems, the demands needed for such are hard to meet in TinyML applications.

3 Problem Definition

As mentioned above (Section 2), new research combine communication and TinyML, especially for updating TinyML models. Collaborative ML approaches have shown to be promising for privacy-preserving data analysis as no raw data needs to be sent to external entities for analysis. Given this, traditional approaches used for model distribution among devices or servers cannot be employed in MCUs as they require a higher order of resources. For example, Secure Aggregation used in traditional Federated Learning (Bonawitz et al., 2016) accompanies complex cryptography to preserve privacy. These cannot be used in Microcontrollers due to the computational complexity and resource overhead. Even if MCUs were capable of such computations, many protocols, including Secure Aggregation, assume that an end-to-end connection can be

achieved. However, this is far from the case.

In a network consisting of MCUs, various environmental factors and the availability of resources in MCUs can make end-to-end communication to a central node or neighbouring nodes impractical. In addition to this, many embedded devices lack a Radio Access Technology (RAT) for communication. For context, out of the few Tensorflow Lite Micro compatible devices, a majority of them have no RATs. Thus, for the few devices that are capable of wireless transmissions, protocols used for transferring TinyML models or other data need to be re-evaluated or restated to suit the constraints and characteristics of an MCU.

3.1 Problem Statement

Due to the lack of resources in embedded devices and the dynamic environments they are placed in, conventional protocols used for transferring ML/DL models cannot be applied.

4 Research Motivation

The flourishing research in TinyML predominantly consists of solutions confined to the limits of an MCU except for a few solutions. IoT, the umbrella topic of TinyML, by definition is a *network* of intelligent objects. However, research with a networking component is picking up the pace, widening the research in the domain of TinyML. Despite the plethora of research opportunities, conventional networking schemes are unsuitable for TinyML as they require higher orders of resources than what can be provided by MCUs. For instance, TCP/IP requires an end-to-end connection, which will significantly reduce the lifetime of a device. Provided the nature of MCUs, connection loss, mobility of devices within an environment can be expected. Thus, providing a reliable and well-optimized model distribution protocol can propel new research and real-world applications of TinyML.

5 Related Work

Citation	Summary	Improvements	Limitations
Doyu et al. (2020)	Propose a dedicated edge/cloud deployed platform which provides TinyML models on-demand.	• First implementation. • Allows cloud participation in TinyML applications.	• Relies on constant communication with the edge/cloud which can affect data privacy, data security and lifespan .

Continued on the next page

Table 1 – *Continued from previous page*

Citation	Summary	Improvements	Limitations
Yesuf and Prathap (2021)	Develop a Probabilistic routing protocol which employs Fuzzy Logic and various social metrics, physical and network contexts.	• Better performance in terms of packet delivery ratio and required overhead.	• Assume nodes are stationary. Due to this, adaptivity of such a solution to dynamic environment is unavailable.
Sanchez-Iborra (2021)	Employs LPWAN in attempts to detach the dependence of a master node in wearables with tinyML.	• Describes metrics used for communication based TinyML solutions. • Provides long range connectivity to wearable devices • Evaluates parallely running TinyML models to understand the limits in complexity.	• Lack of resources to provide sufficient security and constant connectivity to the network make the wearables vulnerable to security and privacy breaches.
Kopparapu and Lin (2021)	Used the Federated Averaging algorithm described in McMahan et al. (2017) to converge TinyML models in a simulated central server.	First implementation of Federated transfer learning in TinyML .	• Use a Serial Bus connection to the central server (computer) which doesn't simulate a real work scenario. • Has a significantly long Round trip time which can result in packet loss.

Table 1: Related Work

6 Research Gap

After conducting a thorough analysis on the existing literature (tabulated in Table 1), the author came to the conclusion that leaving the RAT exposed as Sanchez-Iborra (2021) and Doyu et al. (2021) did can risk data privacy and security while reducing the lifespan of the MCU. Kopparapu and Lin (2021) and Yesuf and Prathap (2021) assume that the nodes at one-hop distance are stationary and maintain steady connections. However, such may not be observable in the real world. The author also observed that despite the abundance of data transferring protocols for low-powered devices, ML model transferring protocols aimed at MCUs with the ability to adapt based on environmental dynamics is non-existent.

Moreover, The lack of optimized, reliable, adaptive protocols in TinyML applications with small footprints allows explorations of untapped research areas. Thus, cognitive network protocols can be developed to fill this gap.

7 Research Contributions

Providing the ability to share models gives space for wider connected applications and research opportunities. Hence, the contributions of this research are as follows:

7.1 Contribution to the Technology

This project aims to develop a Lightweight Opportunistic TinyML Model distributing protocol for compatible Microcontrollers. This is uptaken in hopes to expedite ML/DL model distribution based research and applications. A hypothesis has been made that the proposed schematic for TinyML will adapt and automate the process of manually unloading models from MCUs based on the environment it is placed in.

As a supplementary contribution to the above and the software development domain, the author attempts to define a plug-and-play communication framework as a by-product of the primary research.

7.2 Contribution to the Problem Domain

TinyML relies on resource-scarce MCUs for ML/DL execution. As attempts to update TinyML models by intersecting the cloud/edge paradigms with MCUs is increasing, it is important to discover communication protocols that satisfy the constraints of such devices. The output of this research project will consequently help extend IoT with Collaborative Machine Learning applications and research.

As Chapter 2 mentions, TinyML is a result of fusing the domains of Machine Learning and IoT. Hence, this project contributes to both domains mentioned above under the context of ML-driven-IoT.

8 Research Challenges & Potential

8.1 Challenges

1. **Lack of Resources in MCUs** - The lack of resources in MCUs translates to severe limitations on what can be done/applied within these devices. Additionally, the heterogeneity among devices means that resources vary depending on the manufacturer and use case. Opportunistic networking, however, requires ML/DL algorithms. Moreover, given the research directions, such a networking scheme is likely to be used alongside other ML/DL models, which can take up a significant amount of the available resources. Thus, such a scheme should be incredibly lightweight with little overhead.
2. **Deciding when to transmit data** - Due to their limited ranges imposed by the resource constraints, connectivity among MCUs and other nodes become highly dynamic. The magnitude of dynamism depends on the environment a device is placed in. Hence to preserve battery capacity, decisions on when to transmit data while mitigating re-transmissions circumstances should be carried out.
3. **Environmental factors** - A majority of Embedded devices are deploy-and-forget devices. Given that, the locations that such devices can be placed are limitless. Thus, various environmental factors can affect the use of standardized communication schemes. Thus, making the protocol adapt to its environment in efforts to combat environmental affects on data transmission will be a challenge.

8.2 Potential

Given the compact form factors, the ability to place MCUs in the smallest crevices makes them a great candidate for *deploy-and-forget* (Vilajosana et al., 2010) applications. However, dynamic environments can result in TinyML models being outdated rapidly. To combat this, defacto standard wireless model updates can be employed. Thus, the proposed project will be the intermediary between the MCU and the cloud/edge, facilitating efficient communication. This way, the implications are widespread, which means that such a model distribution protocol can be utilized for Collaborative ML techniques which will lay the stepping stone for new avenues of research.

9 Research Questions

RQ1 - What advancements in Machine/Deep learning can be employed to distribute TinyML models in an energy efficient manner?

RQ2 - In what ways can a cognitive protocol adapt to its environment?

RQ3 - What environmental and network factors need to be considered to develop such a protocol?

10 Research Aim

This research project aims to design, implement, and evaluate a lightweight, adaptive network protocol that automates the process of model distribution for microcontrollers in dynamic environments with minimum human intervention.

This research will elaborate on the process of producing a highly energy-efficient, automated model distribution protocol for TinyML compatible devices. Moreover, the proposed system will be designed to adapt to various environments while running alongside other TinyML models. By doing so, the authors aim to facilitate model sharing which is necessary in Collaborative ML approaches.

11 Research Objectives

Objective	**Description**	**Learning Outcomes**
Literature Survey	A survey on the existing literature is conducted to fulfill the following requirements, • To systematically analyse the existing systems in the domains of TinyML and Delay Tolerant Networks. • To identify the research gaps on the problem. • To determine the technologies, algorithms, frameworks used for developing Opportunistic models. • To identify ways of reducing complexity and size by withholding accuracy levels. • To identify evaluation metrics and criteria.	LO1, LO4, LO6

Requirement Analysis	An in-depth analysis of the requirements was carried out to, • To gather the requirements of domain experts and researchers in order to understand how they will expect it to work. • To get insights on feedback, improvement, and complementary additions through surveys and interviews. • To define the requirements in terms of Software and Hardware through questionnaires.	LO2, LO3
Design	Designing the proposed cognitive networking framework in aims to, • To design the lightweight ML decision component for accurate transmissions based on the network context • To design the mechanism that switches tasks based on context	LO3, LO5
Development	Development of the proposed framework according to the architectures and models developed in the design phase with the identified hardware and software components, • To implement a lightweight decision component that transmits data based on network context • To develop the task switching mechanism • To embed multiple ML/DL models within a MCU	LO5, LO7
Testing and Evaluation	Testing the prototype and getting it evaluated by domain experts/researchers, • To test individual components of the prototype • To evaluate how the system performs in the real world and simulations • To perform validation against the identified requirements for the system	LO7, LO8

Publishing the Findings	Publishing the findings observed in the Implementation, Testing and Evaluation phases of the project, • To validate the final research solution • To contribute to the body-of-knowledge	LO8

Table 2: Research Objectives

12 Scope of the Research

Based on the research objectives, outcomes and existing work, the scope of the project was defined. Here, as the primary research component consists of ML/DL and networking, core features are given priority, whereas non-essential features are given little focus. The scope is defined as follows,

12.1 In-scope

- **A TinyML Model distributing framework for MCUs** – A self-adapting Cognitive framework that is composed of a decision engine which decides when to transmit the data based on its environment and network factors.
- **A smartphone (iOS/Android) application acting as master node** – A smartphone application with the Decision engine of the proposed system that will send and receive TinyML models in a timely manner while being on the move.
- **Adaptivity for the Decision engine** – Providing the Decision engine the capability to adapt based on the environment.

12.2 Out-scope

The components that will not be included in the project are defined below

- By the end of the project the communication framework will not be available for all the TinyML compatible MCUs.
- Threats posed amidst the transmission process are not considered.
- The proposed framework does not include support for devices outside the Arduino TinyML ecosystem of devices.
- The proposed framework will not consider the transmission of data other than TinyML models compiled using the TensorFlow Lite Micro Compiler.

12.3 Prototype Diagram of the proposed system

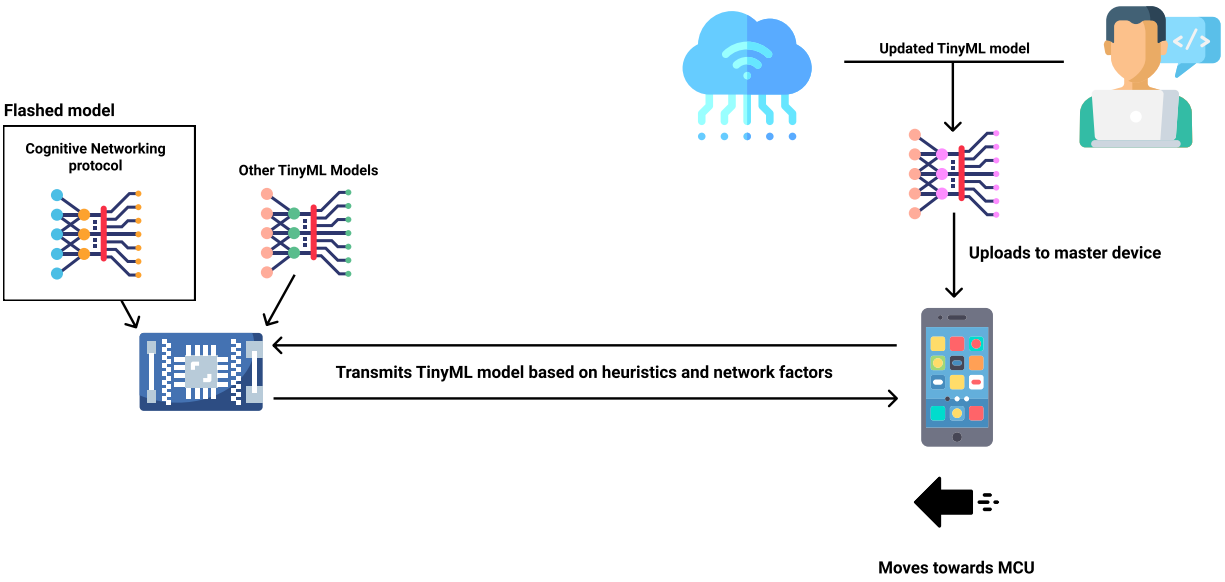

Figure 1: Prototype Diagram of the proposed networking scheme (Self-Composed)

13 Research Methodology

The research methodologies were selected from the predefined Research Onion Model by Saunders et al. (2003).

Research Philosophy	Among the candidate research philosophies pragmatism, positivism, interpretivism and realism, **pragmatism** was chosen. This decision was made through observing the evaluation metrics (E.g. accuracy, latency, etc.), existing research comparisons, and surveys, which consist of both *quantitative and qualitative* data. Additionally, the outcome of this research is to prove the hypothesis with *factual evidence and data*, thus making positivism the best suited philosophy for the proposed research
Research Approach	Out of the available research approaches deductive and inductive, the **deductive** approach was chosen due to the final analysis being a quantitative analysis. This approach aims to prove a hypothesis by trial and error, which in laymans terms is that ML/DL models can be transferred effectively and efficiently with the help of the identified techniques for MCUs.

Research Strategies	The third layer of the onion model is focused on data collection. For this purpose, **surveys** will be the primary research strategy due to the possibility to collect vast but quantitative data. Adding to this, *interviews* will be used as a secondary form of data collection during the latter phases of the project, such as evaluation, testing, and feedback.
Research Choice	Here out of the 3 options mono, mixed and multi methods, the **mixed method** was chosen. To explain, as both quantitative (surveys) and qualitative (interviews) strategies were selected to complement one-another throughout the research.
Time Horizon	The time horizon defines the overall timeframe of the research. Out of the two time horizons, the **cross-sectional** time horizon was selected. In this project, the data will be gathered in a single point of the research – the Requirement Analysis phase. Hence, making cross-sectional time horizon the better suited candidate.
Techniques and Procedures	For data gathering and analysis, mediums such as interviews, observations made through trial and error, similar solutions and literature, surveys and thesis‘ will be employed.

Table 3: Research Methodology

Based on the above research methodologies chosen, the following were established.

Research Hypothesis - Given the resources for wireless communication, MCUs can benefit from an opportunistic, self-adapting and autonomous model distribution scheme.

Research Input - Data to provide the cognitive decision engine with the network context such as signal strength, mobility, etc.

Research Process - Deciding when to transfer a TinyML Model based on the environment and network context.

Expected Research Observation - Observe the decision algorithm transmit TinyML model data (from a MCU) to a moving mobile node (Smartphone) with reduced re-transmission ratio and packet loss ratio.

14 Development Methodology

14.1 Software Development Model

Out of the development models, *Prototype model* was chosen for the research project. To justify, this method has an iterative process of design, trial and error, and evaluation which leaves room for modifications, improvements or new requirements to be accommodated. Moreover, as the evaluation phase is followed after the design and development phase, the improvements required by industry experts and researchers can seamlessly be included by following this model.

14.2 Design Methodology

For the Design Methodology, Object Oriented Analysis and Design (OOAD) was chosen. This is primarily due to the models re-usability and intractability among objects. Additionally, its flexibility is especially useful in a research projects where requirements are prone to change.

14.3 Requirement Elicitation

This sections describes how the requirements for the proposed project will be acquired. For this purpose the following elicitation methods will be used.

1. **Literature Review** - Analysing the similar systems and technologies will provide an understanding of the underlying technologies and algorithms.
2. **Observations** - As the domain of TinyML is a novel research area, experimentation prior to final implementation will be needed. During the experimentation phase, the observations made will be leveraged for end product.
3. **Surveys** - Surveys will be handed out to selected individuals who are sufficiently proficient in the domains on ML and Embedded systems. This is to ensure that the results gathered are of high quality. The data is mostly quantitative, meaning that it can be represented in various graphical forms.
4. **Interview** - Interviews with industry experts will be conducted to gather qualitative data regarding the project. However, since interview data is qualitative, themetic analysis will be required to extract important information from the interviewees.

14.4 Evaluation and Benchmarking Methods

14.4.1 Evaluation

Evaluation of the system is a critical process for identifying whether the proposed solution fulfils the stated problem. Additionally, the process of doing so will explain how well the solution achieved its goals. By reading the existing literature, the following evaluation criteria for the

framework have been pinpointed. They are,

1. **Packet Loss Ratio -** This is a standard evaluation metric used in networking protocols. The proposed framework has a networking component. Hence, the connections between MCUs will be of ad-hoc nature, making them unpredictable. As a result, understanding the Packet Loss Ratio will explain how well the system performs in model transmissions.
2. **Accuracy of decision -** Accuracy is another standard metric used in evaluating ML/DL decision engines. This can provide insights into how well the system performs in deciding when to transmit the model.
3. **Energy consumption -** As MCUs are highly energy-efficient systems, understanding how much energy is used in detecting a nearby fog node until the entire model is transmitted can help make improvements and provide estimations on how such a scheme affects lifespan.
4. **Size of the model -** Unlike conventional resource-rich devices (Computers, cloud computers, mobile phones, etc.), the space needed for the model is essential to evaluate, especially if more than one model will be flashed into a single microcontroller.

14.4.2 Benchmarking

According to the authors' knowledge, no other model distribution schemes have been attempted for microcontrollers. Similar schemes developed for IoT and Mobile phones require orders of magnitude more resource requirements to replicate. Hence, benchmarking the proposed system cannot be done with the existing solutions. However, since an existing model will be transmitted to and from an MCU, a *checksum* can be used as baseline benchmarking to see if the models are equal.

15 Project Management Methodology

Out of the many project management methodologies, the hybrid model **Prince2 Agile** was chosen. Prince2 emphasises on dividing the project into manageable chunks whereas Agile focuses on the iterative cycle (i.e., developing, releasing and getting customer feedback). Moreover, as the author is the sole developer of this research project and since many interim deliverables are expected, the Prince2 Agile method is ideal.

15.1 Resource Requirement

Based on the afore defined objectives, research methodologies and high level architecture, the resources necessary to complete the project are identified. The required software, hardware and data resources with justifications as to why they are required are mentioned below.

15.1.1 Skills Requirements

- **Knowledge on Machine Learning -** As opportunism is a trait of certain Machine learning algorithms. In order to utilize the theories and algorithms, background knowledge on certain opportunistic ML algorithms is needed.
- **Knowledge on Embedded programming languages -** As most implementation of IoT are done through the Arduino IDE, hands on experience with the embedded programming languages will be essential.
- **Designing Skills -** The designing phase of the project is considered to be a milestone. Thus,the knowledge, types of diagrams used for the designing phase need to be revisited.

15.1.2 Software Requirements

- **Operating System (Windows/MacOS/any Linux Distro) -** The resource rich nature of the OS allows to run essential programs such as the web browsers, IDEs required for the project. Additionally an Operating system is needed for converting defined models into MCU compatible code.
- **Python -** Python will be used as the primary programming language to develop the cognitive decision engine for the proposed system. The reason behind this is that Tensorflow Lite Micro primarily uses python for compiling TinyML models.
- **C/C++ -** C/C++ are extensively used languages in developing MCU based solutions. Hence in order to implement the proposed decision engine into a MCU, one of the stated languages are necessary.
- **Android Studio/XCode -** Out of the available IDEs XCode or Android studio are candidate IDEs that will be employed to develop the Mobile application (stated in In-scope in Section 12.1)
- **TensorFlow/TF Lite Micro -** As TensorFlow is a well known ML framework within the community, the supplementary Tensorflow Lite Micro (TFLM) and TFLM Compiler will be used in order to provide ML/DL capabilities to the MCUs.
- **Arduino IDE -** The proposed system will be implemented on an Arduino TinyML compatible MCU. In order to program the MCU as per the requirements for the proposed solution, the open sourced IDE is mandatory.
- **Overleaf -** Overleaf is a SaaS well known within the research community which is used for creating academic documents. Thus, for convenience and ease of use, Overleaf will be used to create reports and documentations.

- **Jupyter Notebooks/Google Colaboratory -** Both are popular environments used for research purposes. Moreover, these will be used as they have an intuitive environment for the trial and error stages of development, and graphical representations of various aspects of the project.
- **Zotero/Mendeley -** A research tool used to manage and backup research papers and relevant artifacts to the local drive or cloud.
- **GitHub/GitLab -** GitHub or Gitlab will be used as a version controlling system to keep track of the programming changes done throughout the project lifetime. Thus, Git is essential to push to a remote repository.

15.1.3 Hardware Requirements

- **Arduino Nano 33 BLE/Sparkfun Edge -** The MCU used to develop, test and evaluate the proposed framework.
- **Intel 8^{th} Gen i Series processor or above -** To perform long running, intensive simulations to train the decision engine.
- **8GB RAM or more -** To be able to manage relatively large volumes of data.
- **Minimum of ≈20GB -** To store relevant datasets, models, applications, and frameworks necessary for development purposes.

15.1.4 Data Requirements

- **Signal Strength Dataset (Sikeridis et al., 2019) -** Signal strength dataset in a time-series format is needed to train the opportunistic decision engine of the proposed framework to decide when to transmit data. The dataset was acquired prior to the initiation of the project.

15.2 Gantt Chart

The authors project plan with the course of actions and respective dates of delivery have been plotted in the Gantt chart below (Figure 2).

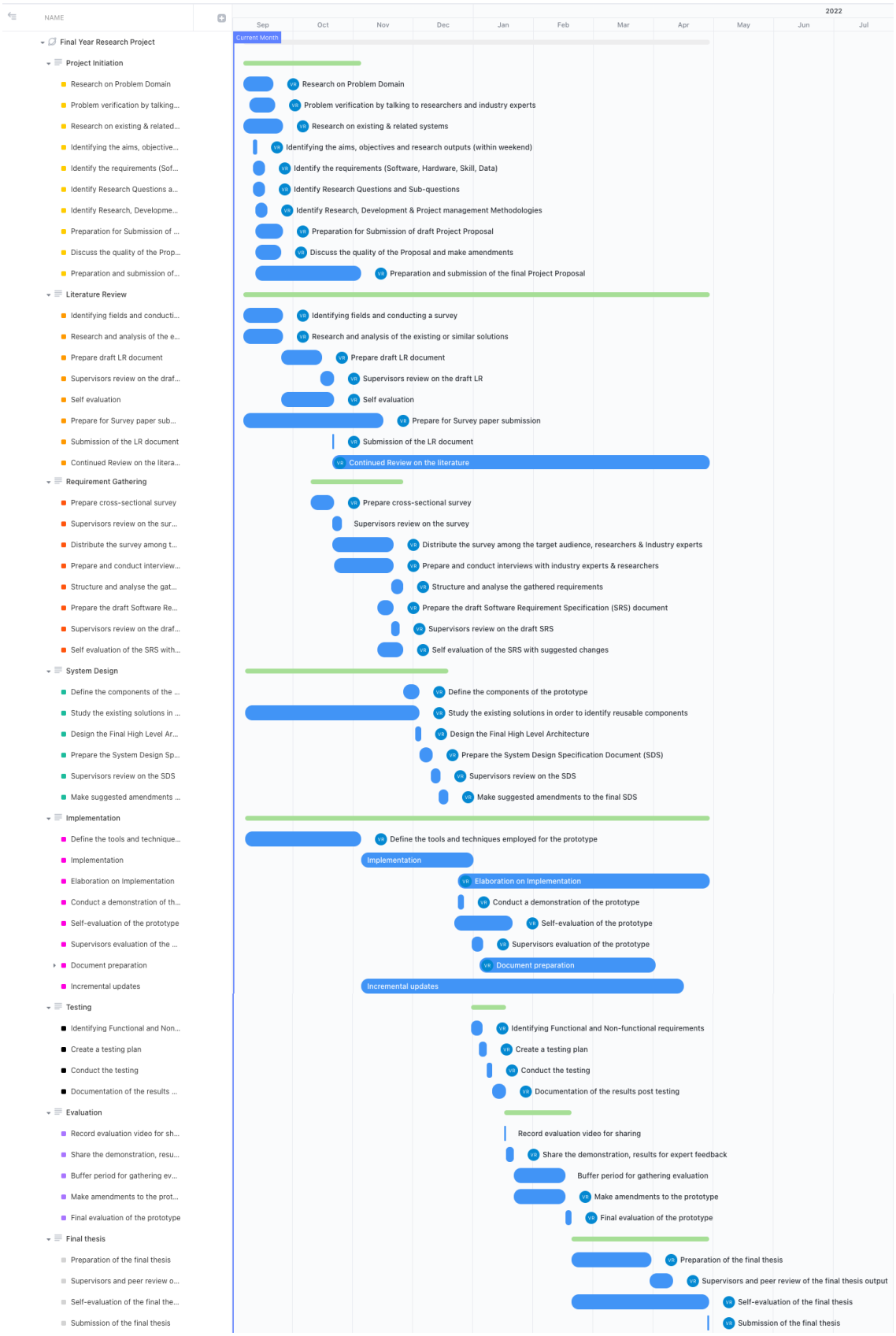

Figure 2: Defined Gantt Chart for the project (Self-Composed)

15.3 Deliverables

Deliverable	Date
Literature Review Critical analysis of the existing work and solutions .	21^{st} October 2021
Final Project Proposal and Ethics form The initial proposal for the research about to be conducted.	4^{th} November 2021
Software Requirement Specification The document that specifies the requirements to be satisfied in order to prototype and collect data.	25^{th} November 2021
Proof of Concept An initial proof of concept of the proposed system.	6^{th} December 2021
Interim Progress Report Describes the progress made on the project till the submission date.	27^{th} January 2022
Test and Evaluation reports Document that presents the evaluation and test findings of the PoC.	17^{th} March 2022
Draft Project Reports Submission of the draft thesis made for the project, in order to get feedback from our supervisors.	31^{st} March 2022
Final Thesis Final thesis submission with documentation on the various aspects of the research project	28^{th} April 2022

Table 4: Deliverables and Dates

15.4 Risk Management

Analysing the potential risks posed by taking on this project can help prepare for such circumstances, if they are to arise. The risks have been identified and tabulated below with the quantified magnitude (0 - 10, where 0 is the lowest, and 10 is the highest), the possible frequency of the authors facing the issue, and finally how such scenarios can be circumvented.

Risk	Magnitude	Frequency	Mitigation
Invalid hypothesis	2	Medium	Continue conducting the research as the final output is considered as a contribution to the research community.
Lack of Knowledge in the domain and frameworks	8	Medium	Get the assistance of domain experts, developers, read blog posts, use StackOverflow, and follow online courses and tutorials.
Lack of resources for the training process	7	High	As training machine learning models require substantial processing power, the processing capabilities of personal computers will not suffice. In this case, an online notebook similar to Google Colaboratory can be used
Arduino MCU malfunction	10	Low	Have a backup device. Possibly a similar model or something capable of running the compiled code.
Lack of space/memory for the decision engine in the MCU	4	High	Reduce the complexity of the Decision engine. Consider using different compression techniques for reducing the overall footprint of the model.
Losing Documentation	9	Low	Backup the contents of the project to the cloud service or use a SaaS platform that automatically backs up to the cloud.

Table 5: Associated risks and mitigation

16 Summary

This document presented the project proposal of the research to be conducted in the upcoming months. Initially, An elaboration to the the Problem Domain and problem definition was conducted. Thereafter the existing systems, challenges of the proposed research were also discussed in

detail. Following that, the objectives and outcomes of the research were discussed where finally, the Research, Development and Project management methodologies were elaborated on.

References

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A. and Seth, K. (2016), 'Practical secure aggregation for federated learning on user-held data'. Available from https://arxiv.org/pdf/1611.04482 [Accessed on 09/09/2021].

Cisco (2020), 'Cisco annual internet report (2018–2023) white paper'. Available from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html [Accessed on 10/15/2021].

Disabato, S. and Roveri, M. (2020), Incremental on-device tiny machine learning, *in* 'Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things', pp. 7–13. Available from https://dl.acm.org/doi/10.1145/3417313.3429378 [Accessed on 07/09/2021].

Doyu, H., Morabito, R. and Brachmann, M. (2021), A tinymlaas ecosystem for machine learning in iot: Overview and research challenges, pp. 1–5. Available from https://ieeexplore.ieee.org/abstract/document/9427352 [Accessed on 07/09/2021].

Doyu, H., Morabito, R. and Höller, J. (2020), Bringing machine learning to the deepest iot edge with tinyml as-a-service. Available from https://www.researchgate.net/profile/Roberto-Morabito-2/publication/342916900_Bringing_Machine_Learning_to_the_Deepest_IoT_Edge_with_TinyML_as-a-Service/links/5f0d54f592851c38a51ce4d0/Bringing-Machine-Learning-to-the-Deepest-IoT-Edge-with-TinyML-as-a-Service.pdf [Accessed on 10/15/2021].

Grau, M. M., Centelles, R. P. and Freitag, F. (2021), On-device training of machine learning models on microcontrollers with a look at federated learning, *in* 'Proceedings of the Conference on Information Technology for Social Good', ACM, pp. 198–203. Available from https://dl.acm.org/doi/10.1145/3462203.3475896 [Accessed on 25th October 2021].

Kopparapu, K. and Lin, E. (2021), 'Tinyfedtl: Federated transfer learning on tiny devices', *arXiv preprint arXiv:2110.01107* . Available from https://arxiv.org/pdf/2110.01107.pdf [Accessed on 10/15/2021].

McMahan, H. B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B. A. (2017), Communication-efficient learning of deep networks from decentralized data, *in* 'Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)'. Available from http://arxiv.org/abs/1602.05629 [Accessed on 10/15/2021].

Ren, H., Anicic, D. and Runkler, T. (2021), 'Tinyol: Tinyml with online-learning on microcontrollers', *arXiv preprint arXiv:2103.08295* . Available from https://arxiv.org/abs/2103.08295 [Accessed on 10/15/2021].

Sanchez-Iborra, R. (2021), 'Lpwan and embedded machine learning as enablers for the next generation of wearable devices', *Sensors* **21**(15), 5218. Available from https://www.mdpi.com/1424-8220/21/15/5218 [Accessed on 10/15/2021].

Saunders, M., Lewis, P. and Thornhill, A. (2003), 'Research methods for business students', *Essex: Prentice Hall: Financial Times* . Available from https://toc.library.ethz.ch/objects/pdf_ead50/4/E57_7072090_TB-Index_005013522.pdf [Accessed on 10/15/2021].

Sikeridis, D., Papapanagiotou, I. and Devetsikiotis, M. (2019), 'CRAWDAD dataset unm/blebeacon (v. 2019-03-12)'. Available from https://crawdad.org/unm/blebeacon/20190312 [Accessed on 16th October 2021].

Tensorflow (2020), 'Tensorflow lite for microcontrollers'. Available from https://www.tensorflow.org/lite/microcontrollers [Accessed on 10/15/2021].

Vilajosana, X., Llosa, J., Pacho, J. C., Vilajosana, I., Juan, A. A., Vicario, J. L. and Morell, A. (2010), 'Zero: Probabilistic routing for deploy and forget wireless sensor networks', *Sensors* **10**(10), 8920–8937. Available from https://doi.org/10.3390/s101008920 [Accessed on 10/15/2021].

Yesuf, F. Y. and Prathap, M. (2021), 'Carl-dtn: Context adaptive reinforcement learning based routing algorithm in delay tolerant network', *arXiv preprint arXiv:2105.00544* . Available from https://arxiv.org/abs/2105.00544 [Accessed on 10/15/2021].

Zhang, Y., Suda, N., Lai, L. and Chandra, V. (2017), 'Hello edge: Keyword spotting on microcontrollers', *arXiv preprint arXiv:1711.07128* . Available from https://arxiv.org/abs/1711.07128 [Accessed on 10/15/2021].