Deploying Machine Learning Models on Resource Constrained Devices: A Case Study

MOHAMMAD UMAR KASHIF, University of Twente, The Netherlands

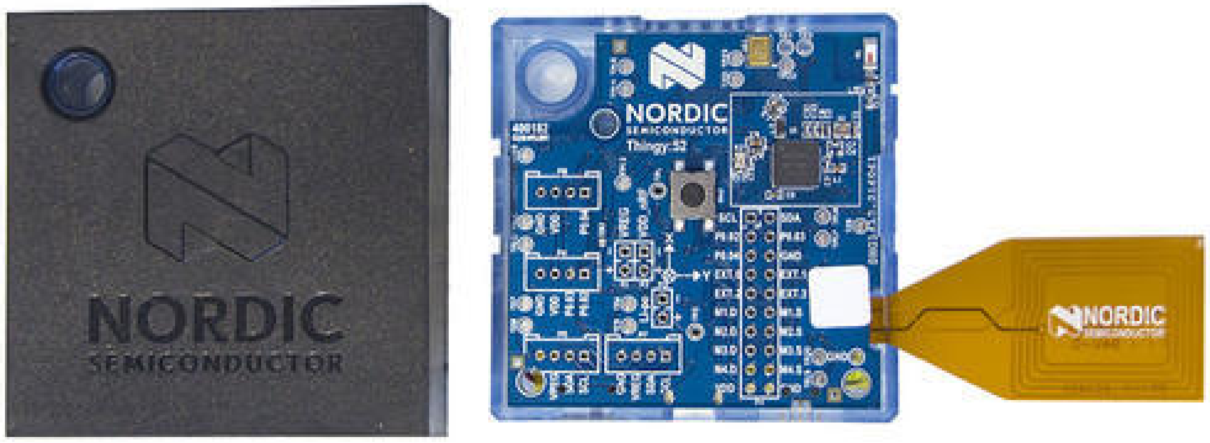

Fig. 1. A Thingy 52 By Nordic Semiconductors. [1]

In the modern day, Machine Learning and Artificial Intelligence Systems have grown exponentially in their capabilities of performing a wide range of tasks, but with that so has their energy demands in the training phase of development and the inference phase on the end device. This has led to severe concerns about their impact on global greenhouse gas emissions. It is not realistic to expect the new era of ML to come to a halt to address these environmental concerns, therefore there exists a need to explore ways to improve the efficiency of these ML models to consume fewer resources. This paper explores some potential improvements to this process, namely deploying Machine Learning models on resource-constrained IoT devices, reducing the amount of data needed to train these models, and minimizing the number of neurons needed to develop them. For the practical aspect of research, we will be exploring the most efficient manner of developing Machine Learning for motion classification on the cloud using Edge Impulse and deploying this model on a Thingy 52, a small IoT Device by Nordic Semiconductors. We will explore the effect of reducing the amount of training data required, number of training epochs, hidden layers, and neurons to converge on an acceptable model despite the reduced training factors and with the limiting resources of the Thingy 52, as well as discussing the various issues encountered and potential future improvements.

TScIT 41, July 5, 2024, Enschede, The Netherlands

Additional Key Words and Phrases: Machine Learning(ML), Edge Computing, Sustainable Artificial Intelligence, Internet Of Things(IoT), Thingy 52, Edge Impulse, Embedded Machine Learning

1 INTRODUCTION

Within the past few decades, the world has seen an exponential rise in the capabilities of Artificial Intelligence systems. From their humble beginnings in the 1960s [13], where systems like the "Shakey" robot and "Eliza" were able to display basic logical reasoning and problem-solving abilities, AI Systems today can demonstrate sophisticated capabilities in perception, reasoning, learning, and creativity. AI systems nowadays surround us in all corners of life, from students using ChatGPT to get help with their assignments, advanced analytical systems predicting future market trends, and expert systems performing medical diagnosis based upon symptoms amongst many more. However, as models become more complex and their usage becomes more widespread, energy demands of these systems have also seen a rather alarming rise [14]. This rise in power usage is concerning, especially at a time when reducing greenhouse gas emissions is critically important for combating climate change [7]. Remarkably, it is estimated that the computational power required to sustain the rise in AI systems doubles every 100 days [5]. While more complex models like Natural Language Processing (NLP) consume significantly more energy [12], for relatively simpler models like image or motion classification it is not necessary to use such energy-demanding sources. While the current state-of-the-art uses a central location where all data collection points send their data to be evaluated, a newer architecture has been explored where data is processed and evaluated at the point of collection instead, without the need of transmitting large volumes of data to a centralized location[8]. This is a process called Edge Computing and will form the primary basis for the research aspect of this paper.

It is evident that there is a need to mitigate the environmental impact of training and deploying AI models. While it is unrealistic to forgo the immense benefits these systems offer in terms of financial power and convenience, exploring alternative, energy-efficient approaches to machine learning is crucial. This thesis investigates strategies for a case study of deploying a motion classification machine learning models on resource-constrained Internet of Things (IoT) devices, aiming to reduce power and energy consumption during the training process without compromising performance by using an Edge Computing architecture.

2 THE ALTERNATIVE

The rise in the capabilities of the "Internet of Things" devices serves as a potential alternative to several of the problems associated with the resource consumption of machine learning. These devices are generally more resource-constrained, therefore deploying Machine Learning models on them requires a slightly different approach to what is conventionally done for larger devices. In particular, due to the smaller amounts of RAM and FLASH available, the models created for such devices are generally kept as small as possible [4]. This is performed by reducing the complexity of the created model to as low as possible. Some of the factors that determine the complexity of these models are as follows;

- *The number of hidden layers in the neural network*
- *The number of neurons in each layer of the neural network*
- *The individual weights and biases*
- *The architecture used*
- *The activation function used*
- *The pruning methods used*

The paper will explore the means through which some of the mentioned points may be minimized for the model to consume as little data as possible during the training phase of the process whilst simultaneously consuming limited resources to be able to perform inference on the IoT Device. The paper will evaluate how changing the collected data for training affects the performance of the model, assess how many epochs it will take to converge on a reasonable accuracy with these volumes of training data, the number of neurons and layers that are reasonable within the case study as well as ensuring that the total resources required do not exceed the capacity of the IoT device.

To connect the research to the goal of sustainability within Artificial Intelligence, as mentioned before it is not realistic to simply reduce consumption of AI systems. While for more complex models (like NLP) it is not feasible to deploy on IoT devices, for a lot of simpler applications it is possible to allocate the task of inference of data to be done on edge devices instead of on a central server. The smaller resource availability of these devices means that the models produced for them are generally more compact, which is often achieved through simplifying the training phase of the model itself initially, reducing power consumption. Although this reduction may be a very small amount in one particular application, the nature of such models (like motion or image classification) is such that they are often continuous, performing up to millions of inferences per second worldwide and a switch to such an architecture on a large scale can potentially lead to a significant reduction in the power and subsequent power consumption of these relevant applications.

3 RESEARCH QUESTIONS

The first item of interest within the research lies within the various challenges encountered while deploying the Machine Learning model on the target device. This includes the entire process of collecting & labeling data, creating a model, deploying this model, and running inference on the target device. We shall also compare how this process differs from the current state-of-the-art.

(1) *What are the technical challenges and performance implications of deploying machine learning models directly on IoT devices, and how do they compare to traditional server-based approaches?*

Another key topic of interest is how the process used to develop and deploy the model on the target device for this particular research may be generalized to form a basis to be used in applications beyond the target scope of this research.

(1) *What are the key components and design considerations for developing a framework to enable the deployment of machine learning algorithms on IoT devices, and how can this framework be generalized for broader adoption and applicability?*

4 ARCHITECTURE

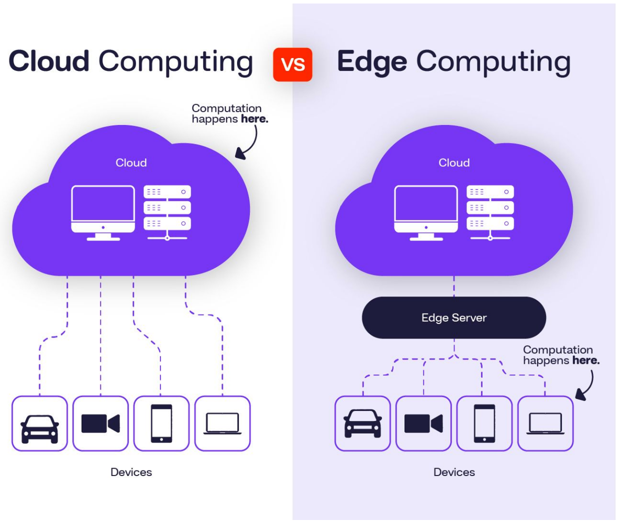

Fig. 2. The difference between a cloud and edge computing architecture [17]

The proposed architecture used in this case study is an example of an application where data is processed near the point of collection (on the Edge) rather than through a central server. Within this research, we will be focusing on creating a motion classification model for a Wii Controller in a Mario game. However, the same architecture may be applied in a wide range of applications where real-time sensor data is collected to perform relatively simple inferences. These applications may include robotics, self-driving cars, industrial processes, etc. The current state-of-the-art is cloud computing, where data is collected by various sensors and sent to a central server to create the model and perform inference [3]. While this architecture is necessary for dealing with larger complex models with high resource requirements, for smaller applications like we have mentioned before we may move the inference part closer to the data collection point [4], as may be seen in figure **??**. Within the context of this case study, data is collected by a sensor on the Thingy 52, and the model is created through the cloud, but the inference data is carried out on the Edge Device itself, rather than through the central server.

Later on, the paper will discuss the various implications such an architecture may bring, explore the various advantages, address the disadvantages, and offer potential future solutions to these.

5 EXISTING RESEARCH

To gain a better understanding of where the industry is currently at in the field of deploying Machine Learning on Edge Devices, I conducted research into various fields to find potential gaps to explore further, as well as finding citations to aid my research. The following list summarizes my findings.

- *A comprehensive overview behind the rise of Edge Computing may be found in a paper by Wang et al. (2020) [10]. The primary point of interest in the context of this project is the various drawbacks associated with traditional cloud computing in many contexts today, with issues arising in reduced bandwidth, latency, security, and privacy. Within the research, the architecture of edge computing is defined and compared to the current state-of-the-art, which will be used to form the structure of the case study in this paper.*
- *Edge Computing and IoT are two different concepts in the field of Machine Learning but have to be combined to form the foundation of the case study. Research conducted by Zhou et al. (2019)[18] outlines how these two concepts are compiled to create the proposed architecture. It also provides detail into the potential of energy efficiency achieved through such means.*
- *A practical example of the use of the Thingy 52 to collect sensor data and transmit information via BLE may be found in the research conducted by Gupta et al. (2019) [6]. The paper outlines their architecture for collecting data from the target device and establishing communication, as well as outlining problems they faced in the particular setup. Both of these elements proved to be useful in the implementation aspect of the research.*

6 THE PRACTICAL RESEARCH

Within the course of this Research Project, the primary focus was on creating a practical example of a Machine Learning Model created by collecting data from an IoT Device, compiling that data on the cloud, and producing a model that may be deployed on a resource-constrained device. The device used to collect the sensor data and run inference on the new data is the Thingy 52 by Nordic Semiconductors [11]. The Thingy 52 is an ideal example of an Edge Device that can collect and transmit sensor data to the cloud while also having the capacity to run inference on basic models, making it suitable for testing the abovementioned architecture. Sensor data is collected from the Thingy 52 and the aim is to produce a model that is able to run motion classification on one of the 5 classes as mentioned below.

- *Left Tilt*
- *Right Tilt*
- *Up Movement*
- *Spin Movement*
- *Still*

Each of these movements corresponds to a move in a Mario game. One of the aims of the project was to utilize the created model to be able to use the Thingy 52 as a Wii Controller for the game, developing on the work produced by an earlier Masters project, a perfect example of using IoT devices as an alternative means to the current state-of-the-art to enhance the efficiency of usage. To run the inference on the Thingy 52, the produced model had to meet the resource restrictions of the device, as seen in table 1. To tackle

TOTAL RAM	256KB
TOTAL FLASH	32KB
USABLE RAM FOR MODEL	128
USABLE FLASH FOR MODEL	12

Table 1. Resources available on Thingy 52

the constraints of these resource requirements, Edge Impulse was used to compile the data and create the model. This was selected as the platform for a variety of different reasons. Edge Impulse is an industry leader in creating models for Embedded Systems, making it much simpler to create efficient models in terms of resource allocation. Furthermore, it is extremely simple to gather and label data using the platform from the Thingy 52. They also offer the ability to use model optimizations like the EON Compiler, which can reduce the RAM & FLASH usage by half while not affecting model performance. Finally, apart from their basic model-making capabilities Edge Impulse also offers a variety of potential efficiency improvements with hardware-related aspects like custom DSP blocks that may further enhance the efficiency of the created model.

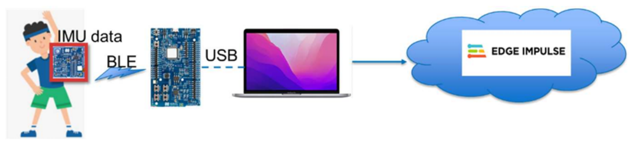

Fig. 3. A visual representation of the architecture

The entire development process is as follows;

(1) *Sensor data (accelerometer data) is collected from the Thingy 52 at a frequency of 50Hz and sample length of 5s*
(2) *This data is transmitted to the Nrf52DK board via BLE*

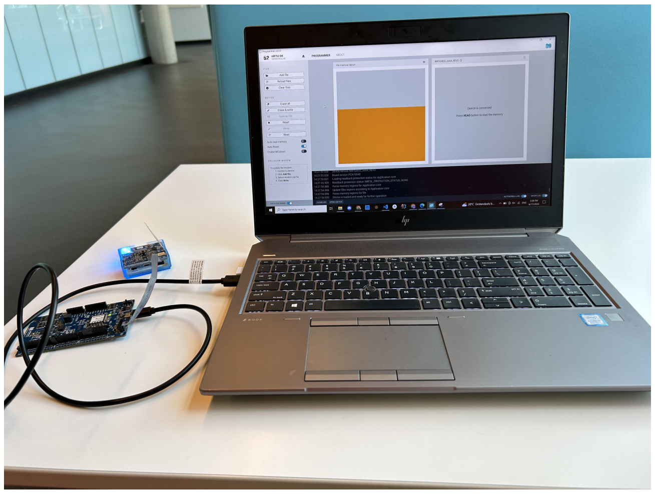

Fig. 4. Connecting the Thingy 52, Nrf52DK Board and the Nrf Connect Desktop Programmer

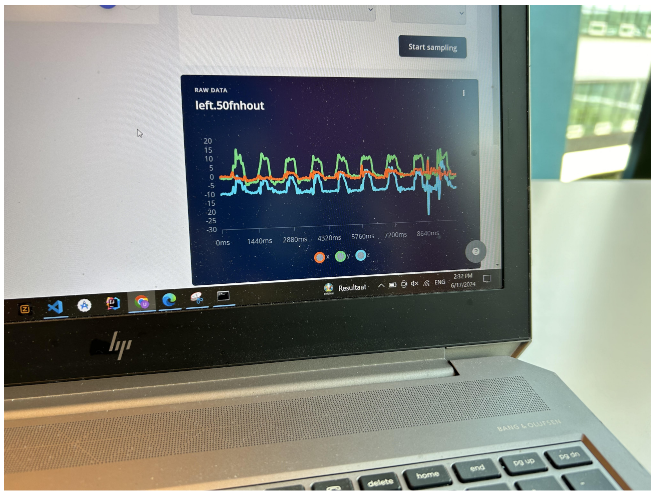

Fig. 5. An example of one of the labeled data, at a frequency of 50HZ and length of 5s with the label "left".

(3) *This data is then transmitted from the nrf board to the desktop computer via USB, as seen in figure 4*
(4) *Edge Impulse Command Line tools (the Edge Impulse data forwarder) are used to transmit this data to the Edge Impulse Project, as seen in figure 3*
(5) *Each of these data samples is then manually provided a label corresponding to the movement detected, as seen in figure 5*
(6) *Once enough data has been collected to create a model, the data is split into a test and training set.*
(7) *Edge Impulse's easy-to-use tools are used to create the features to extract and create blocks to classify the data and detect anomalies*
(8) *A low pass filter is applied to the data and features are generated.*
(9) *The classifier architecture is determined. I can experiment with different numbers of epochs, hidden layers, neurons in individual layers, etc here*
(10) *Anomaly detection is determined. The features selected are usually the RMS anomalies for the x,y and z values.*
(11) *Some values of live classification are selected of varying sample sizes and movements*
(12) *Confidence levels are selected for the anomaly and movement detection. These should ideally be strict to ensure a high level of confidence in the model's ability.*
(13) *The model is then tested to evaluate its efficiency.*

The following is a complete list of hardware and software used to stream the data from the Thingy 52 to creating a model;

- Thingy 52
- nRF52DK Board
- JTAG cable 1.27mm
- nRF Connect for Desktop v4.1.1
- nRF Programmer v3.0.5
- nRF SDK Toolchain Manager v1.2.1
- nRF Connect SDK v2.1.0
- Edge Impulse Command Line Tools
- Edge Impulse Data Forwarder

7 RESULTS

Once the Thingy 52 was connected to the Edge Impulse Project, the practical research commenced. The first point of interest was to investigate the amount of collected data required to converge onto a reasonably accurate model. To investigate this, the first model was created with 10 minutes of total acquired data time, and the accuracy of the produced model was recorded after a selected number of training cycles. After this, a second model with half the collected time (at just 5 minutes) was used and its accuracy at the same number of epochs was recorded. The primary goal of this aspect of the research was to evaluate how the accuracy of the trained model during testing was affected by the total amount of training time fed into it. The accuracy of the testing model was plotted against the number of training cycles to visualize these factors and further research was conducted to evaluate how these results affect the model training resources. To ensure consistency in results, the following elements were kept consistent.

- The split of classes in the dataset was to be equal. Since 5 classes need to be classified, the total split was 2 minutes and 1 minute of recorded data for each class for the 10 and 5-minute models respectively.
- The train-test split had to be consistent. For this application, a 75% to 25% train-test split was selected. While higher than the most common 80& to 20% split, this was selected due to the small amount of data available for the test set.
- All models are trained on just 2 hidden layers, with 20 neurons in the first layer and 10 neurons in the second. This was selected due to the resource constraints of the Thingy 52, deploying greater amounts of hidden layers or neurons increases the model size beyond the capacity of the IoT device. Meanwhile, selecting fewer layers or neurons may result in the model converging at a much slower rate.

- The confidence and anomaly detection threshold were set to be strict to perform a harsher evaluation of the produced accuracy. This was due to the relatively large separation of most of the classified feature blocks. The confidence rating was set to 0.8 (with 1 being the highest) and anomaly rating to 0.3 (with 0 being the highest).
- Since the model has to make live classifications on a video game with a lot of sudden movements, a very high accuracy is required for the model. Therefore, from this point onwards a "reasonable accuracy" for a model refers to an accuracy rate of greater than 90%.

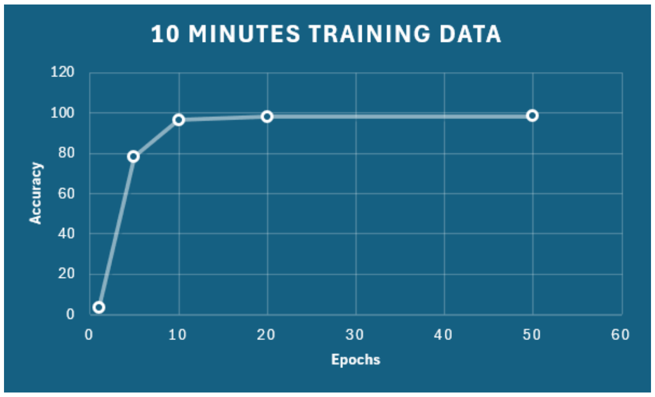

Fig. 6. Accuracy (y-axis) and Epochs (x-axis) with 10 minutes data

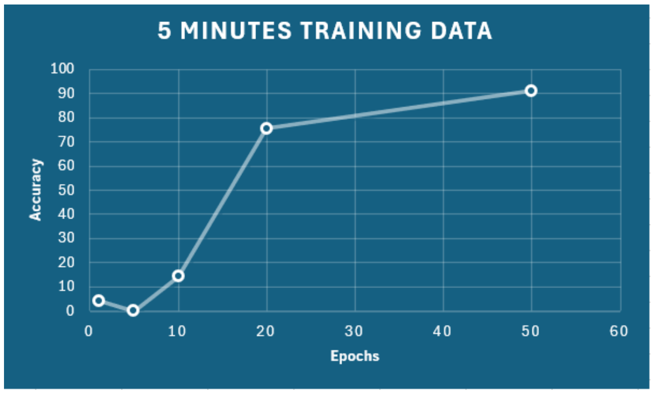

Fig. 7. Accuracy (y-axis) and Epochs (x-axis) with 5 minutes data

The model with 10 minutes of training data converged very quickly onto a model with reasonable accuracy, as seen in figure 6. This was achieved in just 10 epochs. By contrast, it took longer for the 5-minute model to converge into a reasonable margin, as shown in figure 7, with 50 epochs needed to hit the target accuracy. While it may be tempting to believe that reducing the amount of data needed to train the model by half significantly reduces the amount of power consumed in the training phase, a study has shown that in order to produce the most efficient model in terms of power consumption, the number of epochs used has to be kept to a minimum [2]. Therefore, the model that converges to a reasonable model in fewer epochs will be preferred despite their larger dataset size, and the difference in training cycles required to converge on the reasonable model is significant enough to ignore the potential increase due to the reduced training set size. The practical result is backed by another study, which concluded that increasing the training dataset size leads to a logarithmic increase in efficiency per epoch [15]. In context of the wider view of things, it should hold therefore that when selecting the amount of training data required to create models for IoT devices the focus should be on collecting high-quality data to converge onto the target accuracy in as few training cycles as possible, rather than depending on gathering smaller amounts of data relying on more epochs to converge on the target accuracy.

It should be noted, however, that the number of epochs needed to converge to the wanted model in this particular case was rather small due to the nature of the data collected, as seen in figure 8. 3 of the 5 classes are already well separated visually in terms of their features, with the only potential classes with difficult-to-see separation being left & right. This is due to the nature of the movements, the x,y and z accelerometer values within the movements are only similar when turning left and right, while they are vastly different in other movements. This makes the model relatively easy to train with a reduced amount of epochs in the current context, but this may not be the case for all forms of classification, as seen in a simple image classification feature model in figure 9 where different features appear closer together requiring greater processing to classify them.

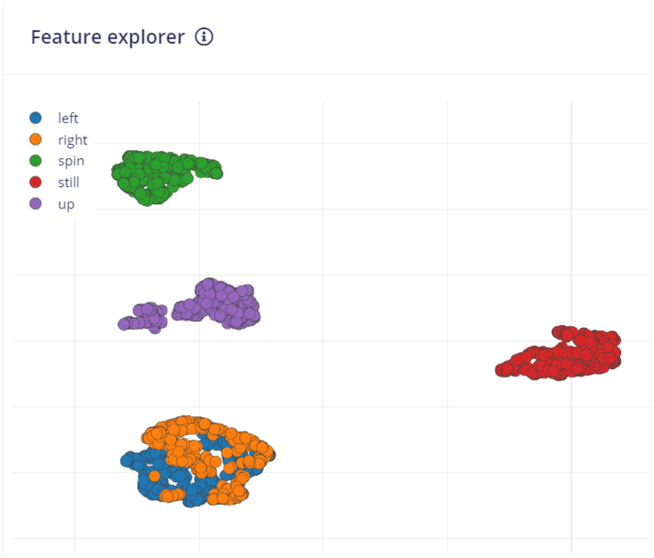

Fig. 8. A visual representation of the features.

With the final model selected with 10 minutes of training data, 50 epochs with 2 hidden layers with 20 and 10 neurons each respectively, a C++ model (quantized by EON Compiler which reduces RAM & FLASH USAGE without loss of performance, as claimed by Edge Impulse) was downloaded. The code from a repository describing how to run Machine Learning on the Thingy 52 [16] was used since a specialist C++ pipeline needs to be utilized as Edge

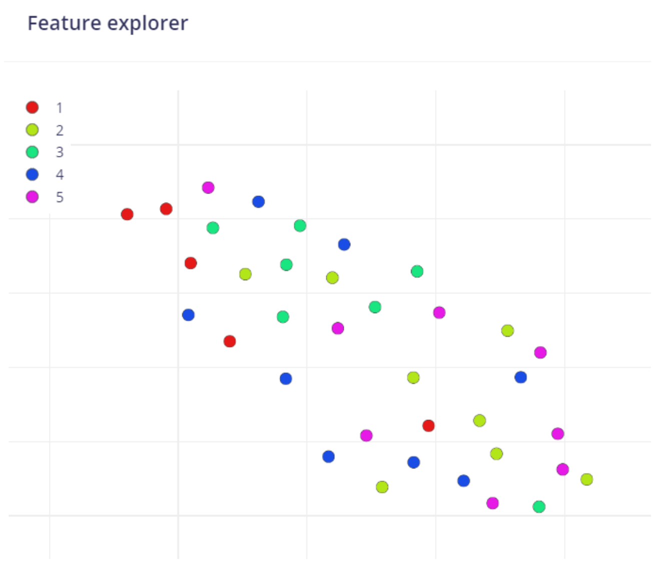

Fig. 9. Feature separation of a small image classification model.

Impulse does not offer direct support for the target device. The total amount of resources available on the Thingy 52, and those used by the model to perform inference, may be seen in table 2. RAM refers to the volatile memory used in the inference phase, FLASH refers to the permanent memory required to run the model while IDT LIST refers to the list of Interrupts that may be generated and need to be dealt with by the code.

Memory Region	Used Size	Region Size	% used
FLASH	372260 B	512 KB	71.00%
SRAM	52672 B	64 KB	80.37%
IDT LIST	0 KB	2 KB	0.00%

Table 2. The resources used to create the model, compared to those available.

The end result is a proof of concept for the proposed architecture, it was possible to develop and deploy a machine learning model with high levels of accuracy within the resource constraints of the Thingy 52 2. Whilst this is just a starting point, it is possible to develop these ideas further and deploy other variants of ML models on edge devices in the future, using the same framework to collect data, create a model, and deploy it that has been mentioned in the paper.

8 PROBLEMS ENCOUNTERED AND FUTURE WORK

Currently, the paper can demonstrate the capabilities of deploying a motion-detecting machine-learning model on a Thingy 52 device. The initial plan was to merge this model with an existing project [9], written by a Master's Student at UT and uses the Thingy 52 as a Wii Controller for a Mario game. Unfortunately, converting the model from C++ into Rust proved to be a task too complicated to complete within the time restrictions of 9 weeks, especially since the C++ model is not directly deployable on the Thingy 52, meaning the entire C++ pipeline provided [16] without support from the original authors. Furthermore, getting the pipeline working to first collect the data and then run the model also seemed more complicated than first thought, since it was written 3 years ago with many of the dependencies being depleted in the newer versions of the nrf connect application, sdk and toolchain, as well as the VS Code extensions leading to a very time-consuming process of trial and error to find the most suitable versions to use (which have been mentioned in the paper). Provided more time, the following would have been completed to add content to the research;

- Converting the C++ pipeline into Rust to allow integration into the Wii Controller project for a more interactive demonstration of the IoT device's capabilities.
- Contacting Edge Impulse to get concrete data into how the resource consumption on their servers varied with the number of epochs, layers, and neurons added.
- While motion detection is a good starting point, Nordic Semiconductors claims on its website that the device is capable of a wider range of IoT applications. Deploying an image classification model within the resource constraints of the Thingy 52 would also serve as a great proof of concept into the capabilities of IoT devices to be able to perform more complex ML Models on Edge Devices.
- Explore the hardware optimizations (like custom DSP Blocks) that Edge Impulse offers.
- The nature of IoT devices is such that they are often mass-produced and, to keep costs low, follow common hardware and software for all produced items. These make them extremely susceptible to security risks since one fault affects a large number of active devices [10]. The scale of production for IoT devices is greater than that for GPUs, CPU, etc since mass production is a key factor of IoT production. Furthermore, due to limited interfaces associated with IoT devices, it provides greater challenges in updating these devices after potential security vulnerabilities are discovered. Further research needs to be conducted on creating a framework in terms of the production of edge devices to address these security concerns.

9 CONCLUSIONS

To conclude, the rise in the capabilities of Machine Learning devices has created an immense load on the energy consumption in the process of training and deploying these models, which potentially leads to tremendous environmental effects in the foreseeable future. While it is not feasible to simply stop in the middle of such a promising era of Artificial Intelligence, there are several means through which we can minimize the environmental effect of Machine Learning by improving efficiency in terms of the training process, of which the number of training cycles and length of gathered data was discussed in this project. Furthermore, with the rise of IoT devices it is also possible to create efficient Machine Learning models on specialised systems for ML on Embedded Devices like Edge Impulse to deploy these models in a more energy efficient manner, like on a Thingy 52 which consumes little energy relative to the current state-of-the-art, providing a framework for developing ML Models for Edge Devices leading to reduced power and resource consumption. While there is great space for further exploration into this topic, this paper outlines the key concepts and acts as a proof of concept into the door of Edge Computing.

10 USE OF AI IN THE PROJECT

During the preparation of this work the author(s) used ChatGPT Copilot in order to generate latex code for creating tables and lists as well as inserting references and images. Furthermore, the citations found in the document for the bib file were also generated with the help of these systems. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work."

REFERENCES

[1] 2017. Nordic Thingy:52. https://www.nordicsemi.com/Nordic-news/2017/06/Nordic-Thingy. Accessed: [insert date here].

[2] Daniel Geißler, Bo Zhou, Mengxi Liu, Sungho Suh, and Paul Lukowicz. 2021. The Power of Training: How Different Neural Network Setups Influence the Energy Demand. *arXiv preprint arXiv:2109.01951* (2021).

[3] Google Cloud. 2024. *What is Cloud Computing?* https://cloud.google.com/learn/what-is-cloud-computing

[4] Rachel Gordon. 2022. *Machine learning at the edge on a microcontroller.* https://news.mit.edu/2022/machine-learning-edge-microcontroller-1004

[5] Simon Greenman. 2024. *How to manage AI's energy demand today, tomorrow, and in the future.* https://www.weforum.org/agenda/2024/04/how-to-manage-ais-energy-demand-today-tomorrow-and-in-the-future/

[6] Karan Gupta, Nitin Rakesh, Neetu Faujdar, and Nidhi Gupta. 2019. IoT Based Solution for Automation of Hospital Activities with High Authentication. In *Smart Systems and IoT: Innovations in Computing*. Springer.

[7] Melissa Heikkilä. 2024. *AI is an energy hog. This is what it means for climate change.* https://www.technologyreview.com/2024/05/23/1092777/ai-is-an-energy-hog-this-is-what-it-means-for-climate-change/

[8] IBM. 2024. *Edge AI.* https://www.ibm.com/topics/edge-ai

[9] Guilherme Issa Lustiano. 2024. thingy-controller: A project for controlling Nordic Thingy:52 devices. https://github.com/guissalustiano/thingy-controller. Accessed: 2024-06-26.

[10] James Maguire. 2024. *What Is Edge Computing?* https://www.datamation.com/edge-computing/edge-computing/

[11] Nordic Semiconductor. 2024. *Nordic Thingy:52 - Bluetooth 5.0 Development Kit.* https://www.nordicsemi.com/Products/Development-hardware/Nordic-Thingy-52

[12] Jos Polfliet. 2024. *You can drive 136,000 kilometers with the energy needed to train one AI system.* https://www.linkedin.com/pulse/you-can-drive-136000-kilometers-energy-needed-train-one-jos-polfliet

[13] Science in the News Staff. 2017. *A Brief History of Artificial Intelligence.* https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

[14] Semiconductor Engineering. 2024. *AI Power Consumption Exploding.* https://semiengineering.com/ai-power-consumption-exploding/

[15] Zhou Shen, Ying Zhang, Wanchun Dou, Jiajun Luo, and Yingshu Chen. 2019. Edge Computing Deployment and Optimization Based on Container Technology. In *Edge Computing: Models, Technologies and Applications*. Springer, Cham, 795–803. https://doi.org/10.1007/978-3-030-15712-8_59

[16] Vinh Tang. 2022. CPS2022_ML: Code and Resources for CPS 2022 Machine Learning. https://github.com/vinhtqc/CPS2022_ML. Accessed: 2024-06-26.

[17] Aron Wagner. 2023. Cloud Computing vs. Edge Computing: Which Is Best for Your Business? *American Cloud* (2023).

[18] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. 2019. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. In *IEEE Internet of Things Journal*, Vol. 7. 6905–6921. https://doi.org/10.1109/JIOT.2019.2951920