SANTA CLARA UNIVERSITY

Department of Electrical and Computer Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

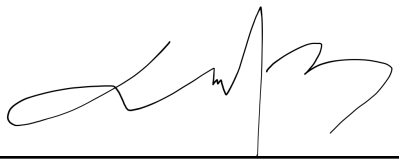
Robb Chun, Anonna Hasan

ENTITLED

Low Power TinyML for Image Recognition

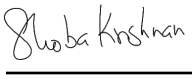
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL AND COMPUTER ENGINEERING



6/14/23

Thesis Advisor(s) (use separate line for each advisor) date



Jun 15, 2023

Department Chair(s) (use separate line for each chair) date

Low Power TinyML For Image Recognition

By

Robb Chun, Anonna Hasan

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical and Computer Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical and Computer Engineering

Santa Clara, California

2023

Low Power TinyML For Image Recognition

Robb Chun, Anonna Hasan

Department of Electrical and Computer Engineering
Santa Clara University
2023

ABSTRACT

TinyML is a rapidly developing field of machine learning that focuses on deploying complex neural networks on to edge devices such as microcontrollers and phones. The goal of this project is to train and deploy an image classification neural network on top of a Raspberry Pi Pico4ML that can be used in an application to assist the visually impaired. In this project, we will use the TensorflowLite for Microcontrollers platform in order to train and convert a convolutional neural network from Keras into a lightweight tflite model that can be deployed to the Pico4ML. Our tasks include training the image classification model on the Cifar10 dataset, modifying the model to run on a smaller subset, and changing the clock speed in order to optimize performance. Our goal is to optimize power consumption and inference speed while maintaining an accuracy that could be deemed safe. By optimizing these aspects of our model, we can create an efficient and meaningful application for the visually impaired .

Acknowledgements

We would like to extend our gratitude to Dr. Hoeseok Yang for both advising us throughout the duration of this project and providing us with hardware and materials necessary to succeed.

We would also like to thank our family, friends, and professors who have supported us throughout our time at SCU.

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Project Statement

The goal of this project is to train and deploy a convolutional image classification neural network onto a low power microcontroller that can be used in an application to assist the visually impaired such as an alert system on a walking stick. Machine learning is a powerful tool that can provide fast and adaptable computer vision solutions for assisting the visually impaired. With TinyML and edge machine learning, we are now able to deploy these complex algorithms onto boards that have only several hundred kilobytes of memory and consume power in the milliwatt power range. Using several optimization techniques such as dataset alteration, clock speed manipulation and dynamic frequency modulation, we will design an image classification neural network that is lightweight enough to run on a small power source such as an AA battery for several days without recharging.

1.2 Project Objectives

The objectives of our project are as follows:

- To train a convolutional image classification neural network on the Cifar-10 dataset
- Perform optimizations on the neural network including dataset specialization, clock speed changes, and dynamic frequency modulation.
- Take measurements for power consumption, inference speed, and accuracy in order to determine the best set of optimizations for both battery life and safety.
- Future work includes designing and producing a physical implementation, trying other microcontrollers, and testing other optimizations such as pruning.

Chapter 2: Background

2.1 Neural Networks and TinyML

The type of machine learning algorithm that we will be using in this project is the convolutional neural network(CNN). A neural network is a machine learning algorithm that is designed to function like the human brain. The neural network is typically composed of one input layer, several hidden layers, and one output layer. The input and output layers are responsible for taking in the input data and outputting the final decision of the network. As shown in figure 1, the hidden layers are composed of nodes that contain weights which determine the decision of the network. A CNN has convolutional layers that are responsible for extracting features from the data such as straight lines in an image, which can then be used to more accurately make a decision. Because of this ability to extract features, CNNs are often used in image classification models like the one we will be using in this project. One downside of CNNs and neural networks in general is their size and complexity, with some neural networks like Resnet50 having over 23.5 million parameters [4].

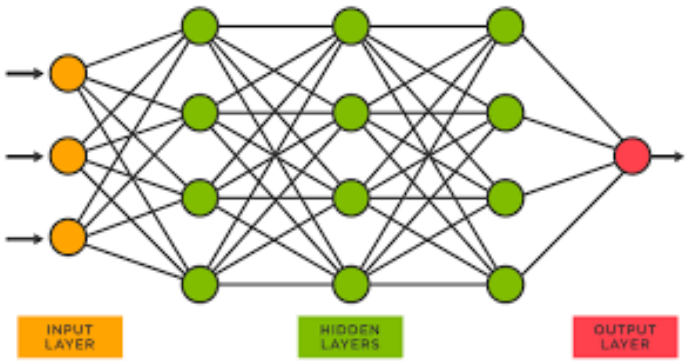


Figure 1: Neural network

In order to deploy complex CNNs onto a memory constrained device like a microcontroller, we utilized the tools from the increasingly popular field of machine learning, TinyML. TinyML focuses on deploying machine learning algorithms onto microcontrollers that consume power within the milliwatt range with a very constrained memory that ranges from a few megabytes down to only a few hundred kilobytes. In addition, TinyML focuses on neural networks that can be deployed locally, and therefore do not suffer the communication latency of a neural network that is running off of a server. Some of the popular platforms for TinyML are TensorflowLite,

Google's platform for TinyML which we will be using, Pytorch, Meta's TinyML platform, and Edge Impulse, a standalone platform specialized in edge AI.

2.2 Current Products and Motivation

Currently, there are not any products on the market that utilize machine learning or neural networks as a means of assisting the visually impaired with walking. There are however, smart walking sticks that use other sensors such as the WeWalk smart cane which uses ultrasonic technology [5]. We believe that by utilizing low power neural networks, we can create an application that is both power efficient, easily deployable, and cost efficient thanks to the design only requiring a microcontroller, which often cost less than one dollar, and a small camera module.

2.3 Hardware and Tools Used

Raspberry Pi Pico4ML: A standard raspberry pi pico board featuring dual cortex M0 chips, 264kB of SRAM and 2MB of onboard flash memory that comes with a HiMax HM01B0 camera and an SPI LCD display specifically for running machine learning examples [6].

Joulescope JS220: A power measurement tool capable of measuring down to a 0.5 nA resolution that can save and display data right to the computer [7].

Chapter 3: Neural Network Structure

3.1 Functional Block Diagram

The design implementation would follow the flow seen in figure 2. The camera module would capture an image from the street and then feed it to the microcontroller. The microcontroller will run the trained neural network and then based upon that prediction, would offer some sort of feedback to the user. This feedback could either be a vibration or an audio cue of some sort that would alert the user if there is something dangerous or potentially harmful in front of them such as a car or a light post.

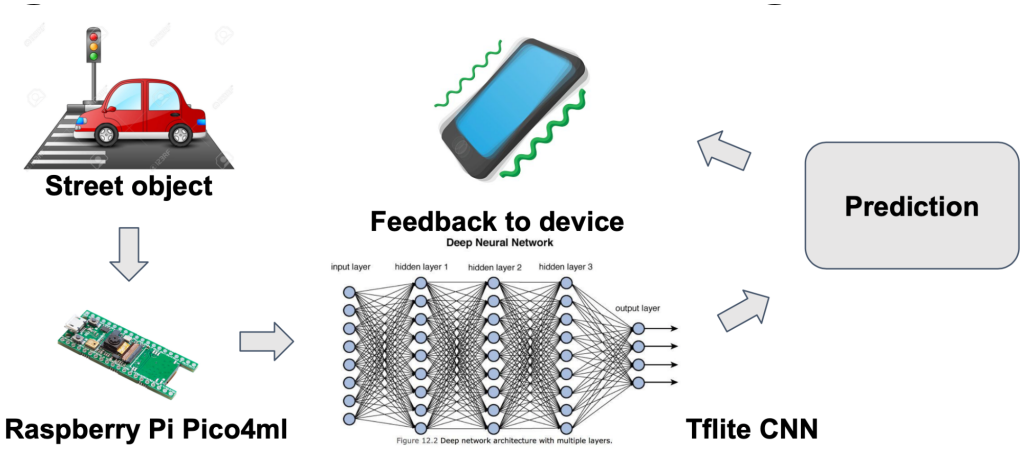


Figure 2: Functional Block Diagram

3.2 Cifar-10 Dataset

The dataset that we used as the base for our image classification model is the Cifar-10 image dataset. This dataset consists of 10 different objects, airplane, automobile, bird, cat, deer, frog, horse, ship, and truck. Each class has 6000 images which are split into 5000 training images and 1000 testing images for a total of 60000 images for all 10 classes. This is a popular public dataset that has been widely used as a standard dataset for image classification models.

3.3 Target CNN Architecture

The neural network model that we are using is a 5 layered convolutional neural network that contains 3 convolutional blocks and 2 fully connected layers. Each convolutional block is composed of 4 components as shown in figure 3. The convolutional layer is responsible for extracting different features from the image. The batch normalization layer standardizes the weight values so that the model is more stable and learns faster. The maxpooling 2D layer downsizes the image dimensions and passes along only the most prominent features to the next layer. This is a common layer that follows a convolutional layer as it helps the convolutional layer extract only the most important information for making a decision on to the next layer. Finally, the dropout layer sets a number of input weights to 0 in order to reduce overfitting which is when a neural network becomes too accustomed to the training data and becomes inaccurate for testing data and actual scenarios. The fully connected layers are responsible for taking the

features from the convolutional blocks and making an accurate prediction based on the features. This model design was selected and tested in order to make a well fit and reasonably accurate model, while maintaining a relatively small amount of weights.

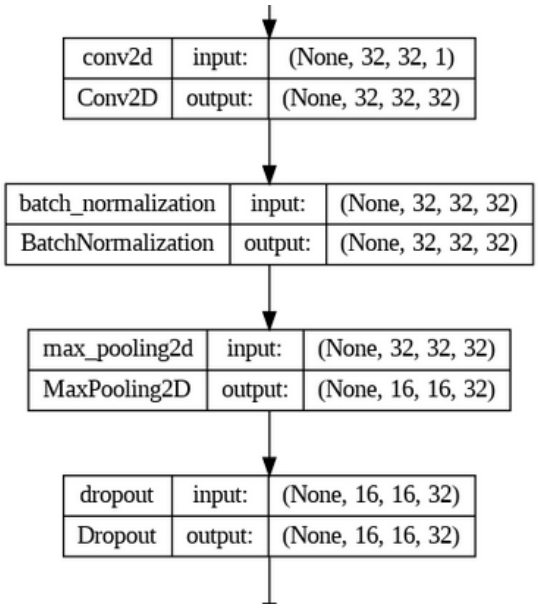


Figure 3: Convolutional layer block

Chapter 4: Model Optimizations

4.1 Test Variables

There are four test variables: accuracy (%), inference time (ms), power (mW), and expected battery life (hours). The accuracy is the accuracy of the convolutional neural network. The inference time is the time it takes for the model to process the camera input data to output a probability score for each class. The power is the power consumption of the board measured by the Joulescope. Lastly, the expected battery life is calculated based on the average current consumption measured by the Joulescope, along with the specifications of two AA batteries that have a current consumption of 5000 milliamps per hour. The following equation was used to calculate the expected battery life:

$$Expected\ Battery\ Life\ (hours) = \frac{5000\ mA/h}{Measured\ Average\ Current\ Consumption} \quad \text{(Eq. 1)}$$

Ideally, through model optimizations, the accuracy will increase to demonstrate that the model is capable of making better predictions. Additionally, the inference time will decrease to demonstrate that the model is able to process data quickly. This is particularly important for the intended use, a walking stick that will be used on the street, because the shorter the inference time is, the faster its reaction time will be, making the product much safer and reliable to use in high-risk situations. To add on, ideally the power consumption will decrease to better accommodate low-power microcontrollers. Lastly, ideally, the expected battery life will increase to make it more efficient and practical.

The following methods were practiced in an attempt to increase the accuracy, decrease the inference time, decrease the power consumption, and increase the expected battery life: reduce the Cifar-10 dataset to a subset with fewer classes, convert the input image data from 96x96x3 images to 32x32x1 images, change the clock speed by reducing it from 266 MHz to 210 MHz, and lastly implement frequency modulation.

4.1 Cifar-10 Subset

The original dataset, Cifar-10, contains the following ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, as shown in Figure 4. There are 5000 images for the train set and 1000 images for the test set. Therefore, in total there are 6000 images per class in total [8].

Figure 4. Cifar-10 Dataset 10 Classes

Since a smaller neural network was used due to memory constraints, it was predicted that a smaller dataset would be more compatible with the model. The main goal of the project is to deploy an image classification model on a low-power microcontroller to create an application that can assist the visually impaired when walking on the street. Since the application is designed for scenarios that one would encounter while walking on the street, a subset of the Cifar-10 dataset with classes that would most likely be on the street was used.

The Cifar-10 subset that was used contained the following classes: automobile, dog, and truck, as shown in Figure 5. Therefore, the dataset was reduced from 60,000 total images to 18,000 total images. The subset has 15,000 images for the train set and 3,000 images for the test set.

Figure 5. Cifar-10 Subset 3 Classes

By reducing the number of classes from 10 to 3, it was predicted that the accuracy would increase and inference time would decrease because the reduced dataset is better fit for smaller models. Typically, it is ideal to have a large training set because a wider variety of data will result in more accurate predictions. However, in this case, it is better to reduce the number of classes to both accommodate for a smaller neural network and to also train the neural network to the specific use of the project.

4.2 Conversions

There were two main conversions needed to deploy the image classification model onto the interface. First, in order to run the model on the Pico4ML, the Keras model had to be quantized and converted to TensorFlow Lite. The Keras model was first converted to TensorFlow Lite. Then, the TensorFlow Lite model was converted to C++ to successfully deploy the model onto the interface.

Additionally, to account for the constraint of the board, the images were converted from 96x96x3 images to 32x32x1 images. First, the images were converted from 96x96x3 images to 32x32x3 images by sampling the input camera data every three pixels. As shown in Figure 6, 96x96x3 images are high-resolution color images. By reducing the pixel sampling rate, the images were converted to 32x32x3 images, low-resolution color images, as shown in Figure 7.

Figure 6. An Example 96x96x3 Image

Figure 7. An Example 32x32x3 Image

Then, the images were reduced from 3 channels to 1 channel. The camera input data was ultimately converted to 32x32x1 images. As shown in Figure 8, reducing the images to 32x32x1 results in a low-resolution, grayscale image.

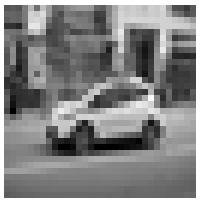


Figure 8. An Example 32x32x1 Image

4.3 Changing Clock Speed

In order to reduce power consumption and increase battery life, the clock speed was reduced from 266 MHz to 210 MHz. It was predicted that reducing the clock speed would reduce the power consumption and increase the battery life because it would reduce the switching rate and make the CPU perform less work in the same amount of time.

However, this did raise a concern regarding the inference time because it was also predicted that the decrease in clock speed would increase the inference time, which is not ideal.

4.4 Frequency Modulation

Lastly, frequency modulation was implemented to decrease the power consumption and increase the battery life while maintaining high-accuracy and a low inference time. The input defines the frequency of the clock speed. There were two frequency modes created: a high frequency (266 MHz) and a low frequency (210 MHz).

The input camera data is used to generate three scores, one for each class. There is a score ranging from 1 to 100 for each class (automobile, dog, and truck). The higher the score is, the more likely it is that the object is being detected. For instance, if the camera is pointed at a dog, the dog score will be high while the automobile and truck scores will be low because the model will detect a dog, but not an automobile or a truck.

The high frequency mode is enabled when one or more of the classes have a score greater than 70 because if any object is being detected, it is necessary that the model is more accurate. Therefore, when any score is above 70, the clock speed is set to 266 MHz.

On the other hand, the low frequency mode is enabled when one or more of the classes have a score less than 70 because if there is no object being detected, it is more important to save power because it is unnecessary for the model to be used at that moment. When all three scores are less than 70, the clock speed is set to 210 MHz.

Combining the high frequency mode and low frequency mode ensures that the power is being saved as much as possible without compromising the accuracy of the model.

Chapter 5: Experimental Analysis

5.1 Measurement Setup

After making all of the changes on the image classification model, three different versions on the board: a 10 image class, 266 MHz clock model, a 3 image class, 266 MHz image class model, and lastly, a 3 image class, 210 MHz image class model were tested using the DC energy analyzer to measure the average power consumption and average current consumption.

As shown in Figure 9, a 5V power supply is connected to the DC energy analyzer. Then, the DC energy analyzer is connected to the Raspberry Pi Pico4ML.

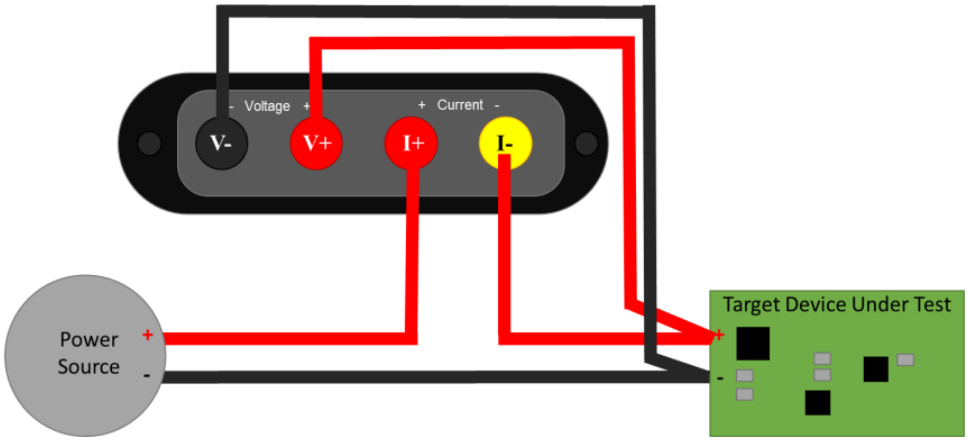


Figure 9. Power Measurement Setup

5.2 Visual Results

The visual results are present on the terminal and LCD display of the board. As shown in Figure 10, the LCD display on the board shows the camera input and the object that is being detected. In this case, the automobile score is the highest compared to the dog score and truck score, so the LCD displays a label titled “car”. As shown in Figure 11, as we feed the camera input into the device, the terminal output shows the score of all 3 classes and the LCD display shows the class that is being detected.

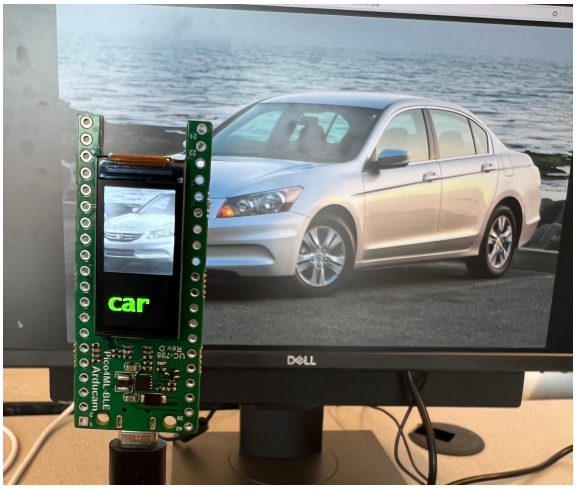


Figure 10. LCD Display

```
capture_frame took 145718 ticks (145 ms)
Display took 16624 ticks (16 ms)
GetImage took 162934 ticks (162 ms)
Invoke took 847938 ticks (847 ms)
automobile score 86 dog score -38 truck score 48
**********
```

Figure 11. Terminal Output

5.3 Measured Results

The accuracy, inference time, average power consumption, and expected battery life results of the model with 10 classes and a clock speed of 266 MHz, the model with 3 classes and a clock speed of 266 MHz, and the model with 3 classes and a clock speed of 210 MHz are displayed in Figure 12.

Table 1. Final Results

Dataset and Clock Speed	**Accuracy (%)**	**Inference time (ms)**	**Power (mW)**	**Expected Battery life (hours)**
Cifar10 (10 Classes), 266 MHz	70	355	346.304	72.018
Cifar10 (3 Classes), 266 MHz	89	315	340.044	73.255
Cifar10 (3 Classes), 210 MHz	89	400	302.801	82.303

As shown in Figure 13, the results demonstrate that as the dataset decreases from 10 classes to 3 classes, the accuracy increases from 70% to 89%. The accuracy remains 89% as the number of classes is kept constant and the clock speed decreases from 266 MHz to 210 MHz.

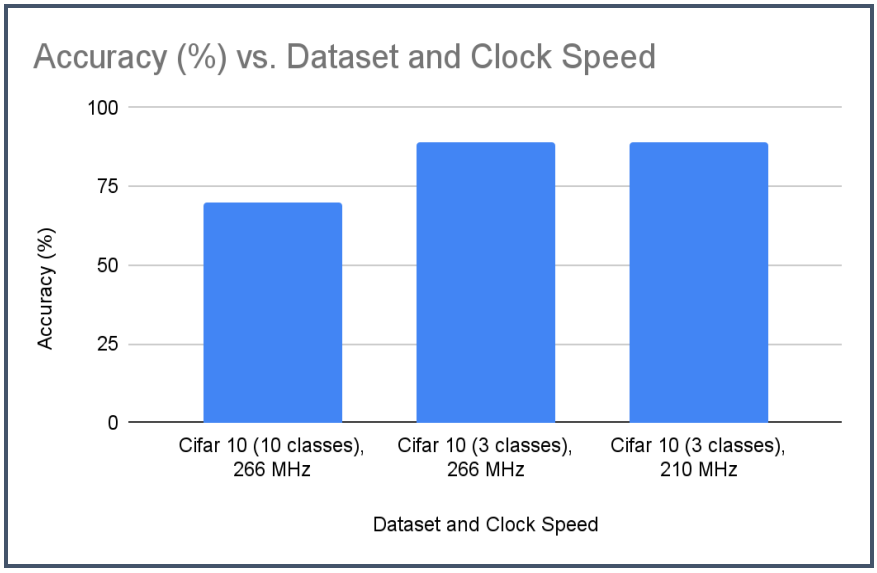


Figure 13. Accuracy (%) vs. Dataset and Clock Speed

As shown in Figure 14, the inference time starts at 355 ms, with a dataset of 10 classes and clock speed of 266 MHz. Then, as the number of classes decreases to 3, the inference time decreases to 315 ms. However, the clock speed is changed to 210 MHz, the inference time increases to 400 ms.

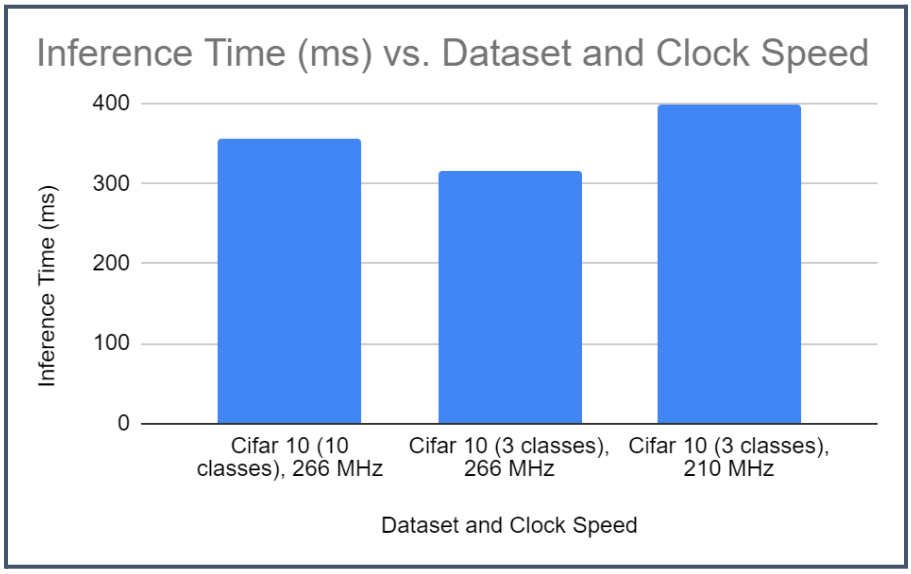


Figure 14. Inference Time (ms) vs. Dataset and Clock Speed

As shown in Figure 15, the average power consumption decreases by around 6 mW as we decrease the number of classes from 10 to 3. However, there is a greater power consumption decrease as we decrease the clock speed from 266 MHz to 210 MHz. The average power consumption decreases by around 37 mW.

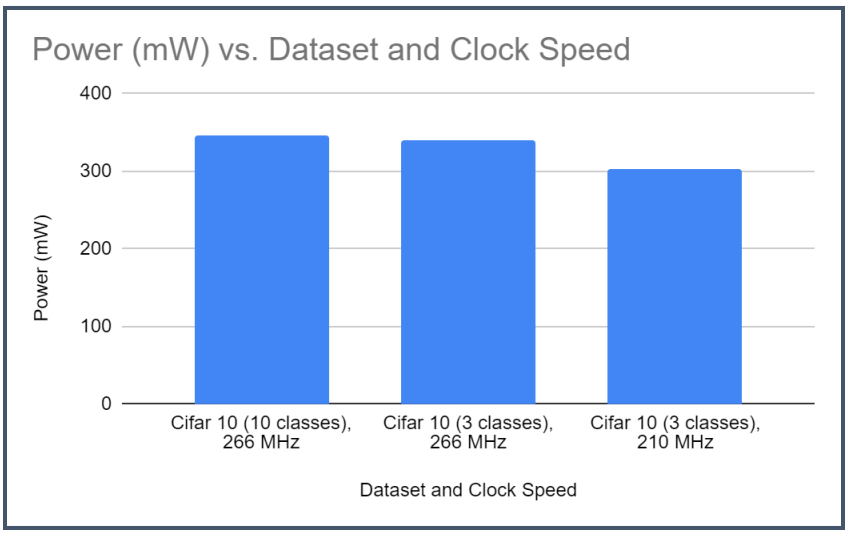


Figure 15. Power (mW) vs. Dataset and Clock Speed

The expected battery life was calculated using the average current consumption and the measurements of two AA batteries. As shown in Figure 16, as we change the number of classes from 10 to 3, the expected battery life increases by a little over one hour. As we change the clock speed from 266 MHz to 210 Mhz, the expected battery life is further increased by around 9 hours.

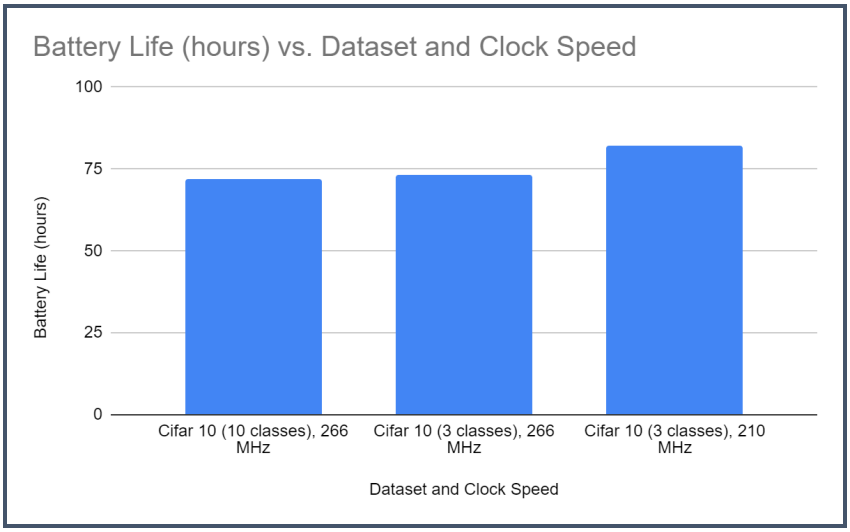


Figure 16. Battery Life (hours) vs. Dataset and Clock Speed

5.4 Analysis of Results

After reducing the number of classes and clock speed, the accuracy increases by 19%, the inference time increases by 45 ms, the average power consumption decreases by 43.503 mW, and the expected battery life increases by 10.285 hours.

The accuracy, average power consumption, and expected battery life displayed ideal trends. However, it is not ideal that the inference time increased. Although the inference time displays a negative trend, the payoff in the expected battery life greatly outweighs the increase in inference time because a 45 ms inference time increase translates to a 10.285 hour battery life increase.

Ultimately, the results demonstrate that reducing the number of classes is an extremely effective method to increase the accuracy while reducing the clock speed is more effective at reducing the power consumption compared to reducing the number of classes.

Chapter 6: Conclusion and Future Direction

6.1 Ethical Concerns

A general ethical concern of image classification models is privacy because there is a risk that the dataset is made up of private information. However, the model used for this project relies on public data for both the training set and test set. Additionally, since the board does not use WiFi, the system does not collect, store, or transmit any public data. This is key for the project's intended use, as it is designed to be used as a walking stick for the visually impaired to use on the street. Since the system does not store any input data, it will not infringe on the privacy of those on the street.

6.2 Future Direction

First, the image classification model can be further optimized by using a dataset that has more classes that cover more objects one would see on the street. For instance, adding classes such as traffic lights, street signs, people, houses, etc, results in a much more reliable and practical product.

Additionally, there is no user interface and the board has not been loaded onto a walking stick yet. This project is not user-friendly yet. Since the product is aimed for the visually impaired, the user interface is another future direction because it is extremely important that the product is designed for the target audience. This requires much more research on accessible technology for the visually impaired community. The product currently does not incorporate aspects such as sound or vibration, so it is necessary to do more research on how to design this product in a way that is easy for those who are visually impaired to use.

6.3 Conclusion

We started this project with the goal of creating a low power image classification neural network that could be implemented on an application to assist the visually impaired. We achieved this goal by taking the Cifar-10 dataset and training a lightweight Keras neural network model on it

and then reducing the dataset in order to create a more specific application. This yielded a significant improvement in accuracy combined with a slower clock speed, we were able to have a model that was both reasonably accurate and lasted an extra 10 hours compared to the base model. However, we do believe that there is future work to be done including creating a custom dataset for our application, further tests on different hardware, and a physical implementation of an actual walking stick that utilizes our neural network.

References

[1] P. Warden and D. Situnayake, *TinyML.* O'Reilly Media, 2020.

[2] J. Lin, W. Chen, H. Cai, C. Gan, and S. Han, "MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning," in *Annual Conference on Neural Information Processing Systems,* 2021.

[3] "Image Classification." Tensorflow. https://www.tensorflow.org/lite/examples/image_classification/overview (Accessed Nov. 12 2022)

[4] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," in *Proceedings of the IEEE Vol. 105,* Dec. 2017.

[5] "WeWALK Smart Cane." Westminster Technologies. https://www.westminstertech.com/products/wewalk-smart-cane

[6] https://www.arducam.com/pico4ml-an-rp2040-based-platform-for-tiny-machine-learning/

[7] "Joulescope JS220: Precision Energy Analyzer." Joulescope. https://www.joulescope.com/products/js220-joulescope-precision-energy-analyzer.

[8] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed Apr. 8, 2009).