CACHE 4 TRASH

Sophia Cho [*1] **Jared Ni** [*1] **Aditi Raju** [*1]

ABSTRACT

Recycling contamination poses a significant environmental challenge, with up to 25% of recyclable waste being contaminated, making it non-recyclable. Current recycling mitigation efforts primarily rely on recycling education, but these approaches rely on extra knowledge and effort from individuals and often result in continued mistakes. In response, we present Cache 4 Trash, a TinyML-embedded automatic waste management system that simplifies waste sorting for users. Utilizing a Nicla Vision inference loop and a motor, Cache 4 Trash employs a TinyML model to classify waste as recyclable or non-recyclable in real-time. The system addresses class imbalances in data, considers edge cases, and navigates the trade-off between accuracy and performance when deployed on tiny chips, leading to a classification of accuracy between 70% and 80%. Our approach enhances both the accuracy and convenience of waste management, offering a promising solution to recycling contamination challenges.

1 INTRODUCTION

Recycling contamination, which occurs when non-recyclable waste is placed in the recycling, causes significant negative impacts on the environment. According to the U.S. Environmental Protection Agency (EPA), as much as 25% of all recyclable waste is contaminated, rendering it no longer recyclable. According to research by Recyclops, a recycling pickup service, the average municipality has a recycling contamination rate of 19% – 40% (Recyclops, 2023). To make things worse, a fear of contributing to recycling contamination may actually cause people to recycle *less* in order to "play it safe." Businesses, for example, can incur contamination fees for misplacing non-recyclable waste in the recycling (RoadRunner, 2023).

Recycling contamination oftentimes comes from well-intentioned people who want to properly sort their recyclable and non-recyclable waste, but have difficulty doing so due to the nuanced differences between the two. To illustrate this difficulty, we list some non-recyclable waste that is commonly mistaken as being recyclable: plastic bags, plastic wrap, flexible packaging (think potato chip bags), cups with wax/plastic, styrofoam (WM, 2023), and plastic utensils (EPA, 2023). Recycling contamination has thus been a difficult problem to tackle.

A popular approach to mitigating recycling contamination is recycling education, usually in the form of brochures and signs (EPA, 2015). This approach, however, still puts the onus on the person who is trying to recycle. Moreover, the brochures and signs tend to contain a lot of information (see Figure 1 below), which makes people still quite susceptible to recycling mistakes.

Figure 1. EPA Brochure on Recycling

We, therefore, introduce Cache 4 Trash, a TinyML-embedded automatic waste management system, where users can simply leave their waste and the system places the waste in the correct (recyclable or non-recyclable) bin. Our hope is that Cache 4 Trash will enable both more accurate and more convenient waste management for all.

2 RELATED WORK

Our project builds on previous work establishing TinyML approaches to environmental problems. One such work is

[*]Equal contribution [1]Department of Computer Science, Harvard University, Cambridge, MA, USA. Correspondence to: Sophia Cho <scho@college.harvard.edu>, Jared Ni <jaredni@college.harvard.edu>, Aditi Raju <araju@college.harvard.edu>.

"Is TinyML Sustainable?" by Prakash et al. In this work, the authors provide TinyML uses cases in tracking greenhouse gas emissions, powering artificial pollinators, and monitoring and adjusting building energy consumption (Prakash et al., 2023).

Our project focuses specifically on the environmental problem of recycling contamination. To that end, there have been other TinyML approaches to waste sorting. "Arduino trash classification TinyML example" by Ahn et al. uses a MobileNet v1 model trained on the Trashnet dataset and deployed on an Arduino Nano 33 BLE Sense and Arducam OV2640 to classify between cardboard, glass, metal, paper, plastic, and trash (Ahn et al., 2020). "Waste Sorting With TinyML" by Varghese uploads a custom dataset to Edge Impulse and runs the EON Tuner for object detection, then deploys the outputted model, which classifies between apples, bottles, and paper, on an ESP32-CAM (Varghese, 2023). "Fight Dirty with Edge Impulse" by Bild similarly uses Edge Impulse to generate a model that detects litter, which is then deployed on a DFRobot Black Gladiator's Raspberry Pi 4 (Bild, 2022). Our project extends these works in the sense that it (1) captures a more nuanced view of the differences between recyclable and non-recyclable waste and (2) automates the waste sorting and disposal process for users.

3 Approach

Cache 4 Trash is a TinyML cache system that sorts between recyclable and non-recyclable waste, automating the waste sorting process for users without adding extra time or effort to their usual habits. We opt for a TinyML approach to smart waste classification because our microcontroller is "always-on." When there is no trash on the cache, the embedded ML model, using an always-on Nicla Vision camera, produces a classification of "No Trash," which keeps the cache lid level. When a user places trash on the cache, the model, again using the Nicla Vision camera, produces a classification of either "Recyclable" or "Not Recyclable." Depending on this classification, the cache lid tilts toward either the recyclable bin or the non-recyclable bin.

3.1 Hardware

One of the main hardware components we used is the Arduino Nicla Vision. Using Nicla's always-on capabilities, we can create a computer vision feedback loop that captures the field of vision of the Nicla when attaching it to the ceiling of the cache system, effectively capturing any trash the user may add to the cache. The information is processed immediately by our backend quantized firmware classification model, where a tuple of probability is generated, identifying the item in the field of view into three categories: no trash, recycling trash, or non-recycling trash. If no trash is present, the firmware continues the feedback loop process and takes more snapshots; if there is trash in the field of view, then the category with the highest probability, given an increased threshold favoring non-recycling trash (since it is better to classify recycling trash as non-recycling trash than vice versa), is selected, and the boolean signal activates one of the two containerized LED lights next to their own photo-transistor, respectively, which detects which light has been activated.

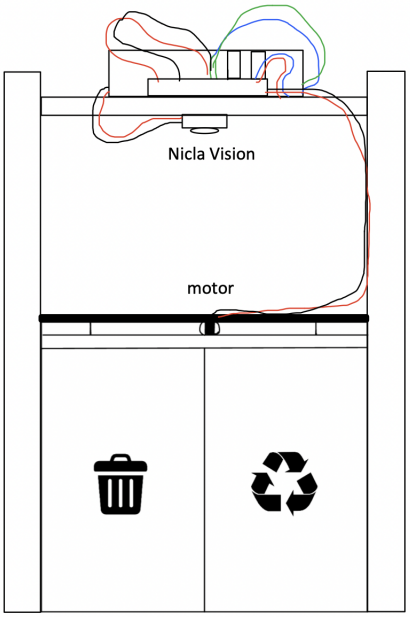

Figure 2. Cache 4 Trash Structure

The photo-transistors, attached to another microcontroller (an Arduino MKR Zero), detect any LED signal in a feedback loop and continuously feed the information to the Arduino MKR Zero. If either of the two LED signals is detected, then power to an H-Bridge attached to a battery pack and a motor will be activated in the direction according to which LED has been turned on. The motor's activation in either direction allows the turning of the electric seesaw to correctly place the trash into either the recycling or the non-recycling can.

3.2 Data Collection

Curating the right dataset was an essential step in building an effective model for Cache 4 Trash. There are already several existing datasets on waste classification, such as Trashnet by Gary Thung and Mindy Yang, Waste Classification data by Sashaank Sekar, and Drinking Waste Classification by Arkadiy Serezhkin. These datasets, however, were too general-purpose for our specific system. We chose to collect our own data so that we could better model real-world use

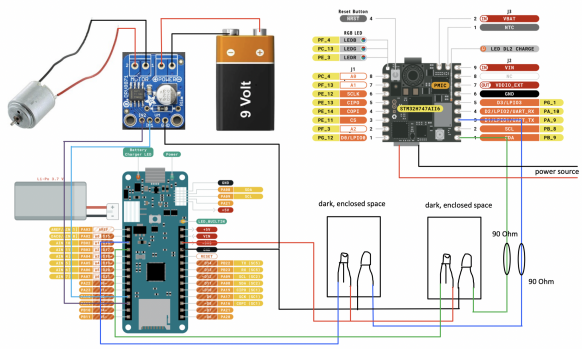

Figure 3. Hardware Circuitry (see GitHub for details).

cases, have greater flexibility to handle edge cases, and use data that matches our system configuration. Cache 4 Trash classifies waste using a Nicla Vision camera at 21 inches above the black cache, so we created our own dataset that reflected all of these settings. That is, we took images of different types of waste against a black background at a birds-eye view of 21 inches using a Nicla Vision camera.

The following is a breakdown of each type of waste in our dataset, along with a visualization of the size of our classes.

Recyclable	Non-Recyclable	No Trash
Glass Bottles	Plastic Bags	Nothing
Glass Containers	Plastic Wrap	
Papers	Flexible Packaging	
Plastic Bottles	Cups w/ Wax/Plastic	
Cardboard	Styrofoam	
Tin, Steel Cans	Plastic Utensils	
Aluminum Foil	Mixture	

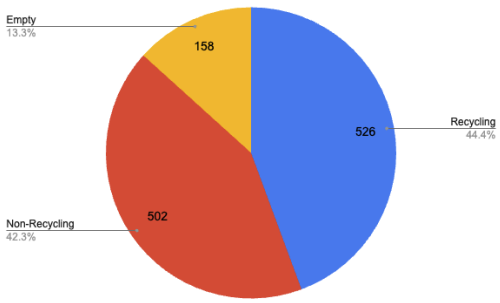

Figure 4. Dataset Breakdown

By the end of our model-building process, our dataset consisted of 1,468 images, with 526 in the recycling, 502 in the non-recycling, and 158 in the empty classes. We manually split this dataset to have a 81-19 train/test split. We opted to manually select which images to move to the test dataset so we could ensure that each category within each class (eg., tin cans) was equally represented.

3.3 Model Building and Tuning

We built our model by uploading our data to Edge Impulse and running EON Tuner for image classification, with the Nicla Vision as our target device. The highest accuracy a model on the EON Tuner reached was 94%, by applying transfer learning using MobileNet v2. However, the

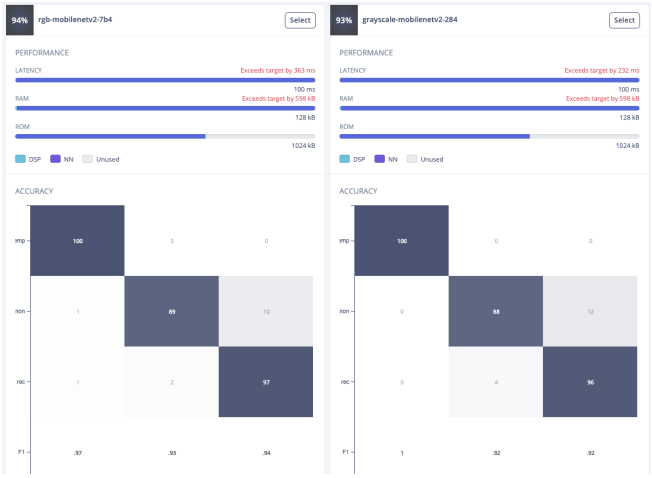

Figure 5. EON Tuner 1

best result from the EON Tuner was too large to be deployed on a Nicla Vision even after quantization due to its need for excess RAM during and after transfer learning. Thus, we picked a multilayer perceptron classifier model that had a relatively high accuracy of 77.7% on validation test sets, 73.05% on the test dataset, and also was memory-constrained enough to be deployed onto Nicla Vision. Additionally, we collected more data to reduce the class imbalance in the classification, and in addition, wrote a data augmentation script to increase the diversity of data and prevent overfitting.

We tried both parameter tuning with and without first running the data augmentation script to increase diversity in the dataset. Given the same multilayer perceptron architecture, the performance and accuracy of the augmented and unaugmented dataset models are nearly identical, though inference time is halved.

3.4 Model Deployment

In order to deploy the model on the Nicla Vision, we quantized it from using float32 parameters to int8 parameters. The quantized model used 4 times less RAM than the non-quantized version (215K vs. 849K) but still maintained very similar accuracy. The quantized model also used less RAM and had lower latency than the non-quantized version.

Figure 6. Test Accuracy of Data Augmented Model

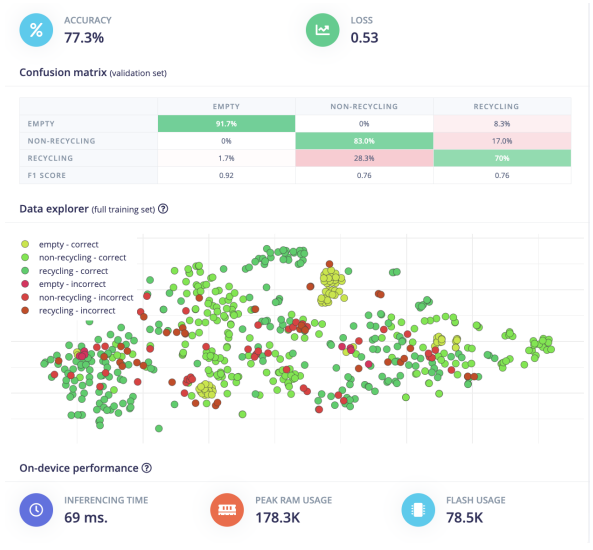

Figure 7. Test Accuracy of Unaugmented Model

Model	Latency	RAM	Flash	Acc.
Quantized (int8)	114 ms.	215.4K	98.1K	72.70%
Unoptimized (float32)	276 ms.	848.5K	245.1K	73.05%

Figure 8. Quantized vs. Non-Quantized Models

4 INSIGHTS

4.1 Class Imbalance

Throughout this project, we learned how class imbalance can greatly affect the accuracy of our model. We first encountered this in the early iterations of our model: we had very few images of the empty class compared to the recycling and non-recycling classes. This significantly reduced the accuracy of our model as it struggled to differentiate between dark pieces of trash (such as black plastic utensils on the black background) and empty backgrounds. We, therefore, took more photos of empty backgrounds to increase the size of our empty class. This reduced class imbalance and improved model accuracy to the point where it could recognize black plastic utensils on a black background.

We also had class imbalance in terms of our non-recycling class. Our initial dataset included significantly more images for the recycling class than for the non-recycling class. This resulted in more non-recycling images being labeled as recyclable, which would further increase recycling contamination. Thus, we added more data to our non-recycling class, including flexible packaging and plastic utensils.

4.2 Addressing Edge Cases

As we developed our model, we ran into an increasing number of edge cases that we tried to account for.

First, different lighting conditions could affect the accuracy of the trash classification model, so we wrote a data augmentation script that added new images to the dataset with a large range of lighting conditions. Additionally, users could throw a mix of recyclable and non-recyclable materials together, in which case the system should throw the trash into the non-recycling bin. To address this, we added images to our dataset consisting of a mix of different materials, including multiple recyclable materials, multiple non-recyclable materials, and a mix of both.

We also considered the real-world implications of Cache 4 Trash. It is more harmful if a non-recyclable item is classified as recyclable rather than the other away around. So, we added an uncertainty check to our model where if the model is uncertain about which category trash belongs in, it will tilt the lid towards the non-recyclable bin.

Despite all these considerations, the model alone may fail to account for further edge cases. For this reason, we added an override option. In this case, if users press either of two buttons (corresponding to recycling and non-recycling), they can override the model's classification and tilt the lid in the correct direction. Ultimately, we learned that a model alone cannot account for all possible edge cases. Combining TinyML with another form of user input allows for more effective integration of models into the real world.

4.3 Model Variability

The models we trained and examined vary drastically in performance, accuracy, and other related constraints. Using the EON Tuner on Edge Impulse, we were able to find models that perform extremely well on our dataset, with an accuracy of 94%. However, high accuracy requires a high trade-off in terms of performance on edge devices. The 94% accuracy model utilizes transfer learning from MobileNet v2, exceeding Nicla Vision's usable RAM by 598 kB. The model we were able to deploy onto hardware, with a post-quantization accuracy of 72.70% and a 215.4K RAM usage, is the highest accuracy model that is within the Nicla Vision's RAM requirements.

Indeed, the trade-off between accuracy and performance is one of the biggest lessons we've learned through this project. With a validation accuracy of 77.7%, our deployed model's accuracy suffered loss after quantization, yet quantization reduced the RAM usage by nearly 4 fold. The best models apply the latest advances in transfer learning, utilize state-of-the-art model architectures, and have many more neurons. Yet, no matter how great and accurate the model is, we cannot use it to achieve our stated goal if we cannot deploy it onto memory-constrained devices. Thus, the variations in model accuracy that we observed are the gap between theory and implementation in TinyML, where theory is constrained by the harsh confines of reality.

4.4 Hardware Limitations and Model Choice

Our decision to limit the project to using the Nicla Vision and the choice of an MLP network was so that we could focus less on hardware issues or raising model complexity and more on data engineering. Inspired by the data engineering papers we read in class, we avoided adding further complexities and spent the majority of our energy on collecting and using the highest quality data that is as fitting to our environment as possible.

Having learned how to use Nicla Vision in previous class projects, and given the extensive documentation of working with Nicla Vision in the context of machine learning on Edge Impulse, we decided that by utilizing the always-on feedback loop that Nicla Vision is built for, we can abstract away many of the hardware details of working with micro-controllers for doing computer vision tasks and rather focusing our energy on data engineering (since CS 249r is, after all, a ML class). Yet, we still experienced issues with the hardware, not being able to establish protocol communication between the two Arduino micro-controllers that we used and running into difficulties with using the motor due to lack of access to a motor driver. For our prototype project, hardware mechanisms are simply a way to enhance the usability of our model, so data quality is what we focused on.

The choice to use an MLP network is straightforward. We had the choice of utilizing MobileNet transfer learning, though it would significantly increase the model size if we used MobileNet v2, and MobileNet v1 performed worse than vanilla classification models in terms of accuracy, potentially due to the specialized environment of our cache system and the benefits of minimizing noise in our data collection (e.g., by utilizing black backgrounds). MobileNet v1 might have had limited neurons compared to a non-transfer learning classification model and was under-fitted to our specific data setting as a result.

5 Conclusion

In conclusion, Cache 4 Trash is a promising solution to the pervasive issue of recycling contamination. Relying on users to self-sort trash often results in inaccurate classification and extra effort. Cache 4 Trash, with its innovative TinyML-embedded automatic waste management system, shifts the paradigm by abstracting waste sorting from users and simplifying the process. By leveraging the power of edge devices and TinyML, our approach demonstrates real-time waste classification, addressing challenges such as class imbalances and edge cases. Through accuracy-performance trade-offs, data engineering, and model tuning, Cache 4 Trash is a step forward in reducing recycling contamination and securing a greener future.

6 Contributions

All group members contributed to data collection and engineering. Sophia Cho contributed to gathering and improving categories of image data, while Jared Ni and Aditi Raju contributed to engineering the hardware components, circuitry, and coding the inference mechanism. All group members worked on building the model. All group members contributed to writing this paper, constructing and presenting the group presentation, recording the project video, and creating the project code base.

References

Ahn, M., Koo, J., Kim, J., and Kang, C. Arduino trash classification TinyML example. 2020. URL https://github.com/lightb0x/arduino_trash_classification?tab=readme-ov-file.

Bild, N. Fight Dirty with Edge Impulse: Autonomous Litter Detection Robot. URL https://edgeimpulse.com/blog/fight-dirty-with-edge-impulse-autonomous-litter-detection-robot, 2022.

EPA. Confronting Contamination. 2015. URL https://www.epa.gov/sites/default/files/2015-12/documents/cashwell_.pdf.

EPA. How Do I Recycle Common Recyclables. 2023. URL https://www.epa.gov/recycle/how-do-i-recycle-common-recyclables.

Prakash, S., Stewart, M., Banbury, C., Mazumder, M., Warden, P., Plancher, B., and Reddi, V. J. Is TinyML Sustainable? *Communications of the ACM*, 66(11):68–77, 2023. ISSN 0001-0782. doi: 10.1145/3608473. URL https://doi.org/10.1145/3608473.

Recyclops. Understanding Recycling Contamination. 2023. URL https://recyclops.com/understanding-recycling-contamination/#:~:text=According%20to%20the%20EPA%2C%20as,be%20recycled%20can%20be%20frustrating.

RoadRunner. What Is Recycling Contamination? 2023. URL https://www.roadrunnerwm.com/blog/what-is-recycling-contamination#:~:text=Did%20you%20know%20that%20a,contamination%20fee%20as%20a%20result.

Varghese, A. Waste Sorting With TinyML. 2023. URL https://www.hackster.io/arunvarghese/waste-sorting-with-tinyml-7af7b1.

WM. Recycling 101. 2023. URL https://www.wm.com/us/en/recycle-right/recycling-101.

A APPENDIX

Github Repository: github.com/AditiR-42/cache-4-trash. This repository includes the dataset we created for this model, the firmware to deploy the model on Nicla Vision, and the code to classify trash in real-time.

Demo Video: youtu.be/d683X9GW1_0

All group members have collectively agreed to allow the publication of our final report video on YouTube by the course staff.