Image Classification for Grocery Store Items: TinyML Applications in Retail

Jessica Edwards
Harvard University
jedwards@college.harvard.edu

Fatima Mbaye
Harvard University
fatoumatambaye@college.harvard.edu

Abstract—Using the Freiburg Grocery Dataset, our research aims to detect grocery images in 10 classes and build ML models able to be deployed on embedded systems. Using Edge Impulse, a leading development platform for machine learning on edge devices, we built several models using transfer learning and neural network architectures to evaluate their memory consumption and performance on our 10 chosen classes. We achieve better accuracy using the transfer learning models than for the original neural networks, and address the constraints, trade offs, and future work that could result from our models.

I. INTRODUCTION

Tiny Machine Learning has a lot of potential in the retail space; with its low cost and memory usage that can be implemented in several image classification models, TinyML in retail can be used to inform storekeepers if a shelf is out-of-stock of a product, help elderly or visually-impaired shoppers read brand names on products they're interested in, and implementing automatic checkout at a low cost similar to Amazon Go. There are clearly several applications for grocery image detection and computer vision in stores that benefit both stores and consumers, and while grocery items image classification is a well-researched problem, there is a lot of opportunity to tackle this problem more in the TinyML space. This problem is ripe for a TinyML solution because TinyML is less expensive than other resource tracking methods using machine learning, given that it does not require as much power and the device and training materials are relatively cheap. TinyML would also allow retailers to be more efficient in managing and selling their products, resulting in increased savings and earnings for the retailer. Getting TinyML technology into this problem space would also benefit the field itself because it would increase the amount of stakeholders in the TinyML space and encourage more activity, research, innovation and collaboration. However, we do acknowledge that our solution has its constraints, including spatial constraints because the application cannot save specific items in need of restocking indefinitely, and object identification because it would be difficult to train the model to recognize all possible items that can be sold in a store. Despite these constraints, we still believe that this application of TinyML would be useful in the long run and believe that our research project is a first step towards deploying more TinyML grocery image detection solutions in stores.

II. RELATED WORK

A lot of research has been done in the general space of object recognition within the general field of machine learning, but much less has been done for object and product recognition in the retail space, especially in the tinyML space. There are numerous reasons for the lack of work in this space, and part of it has to do with the lack of publicly available datasets that are complete and the vast amount and diversity, not to mention the quick turnaround time, of products offered at even the smallest grocery stores [1]. There is also the issue of flexibility and the need for the ability to add new classes with minimal or no retraining without forgetting previous classes, something neural networks are not necessarily known for [1]. This has meant that there are many separate works that focus on completely different aspects of retail product recognition, and that there has not been much consideration of this subject within the TinyML space other than that it is a space with many opportunities for it.

Reference [2] introduced a dataset of 4,947 images of grocery items in various cluttering and lighting conditions collected in a real-world setting. The baseline proposed in [2] is a mean accuracy of 78.9% achieved with a deep neural network classifier with the CaffeNet model architecture trained with the images from the 25 classes of the dataset. They found that their neural network did better classifying images in the candy and jam, and pasta categories, despite the variations in packaging, lighting, and number of items in the image but did poorly when faced with items with plain white packaging, misclassifying them as flour, and items with misleading packaging such as juice or cereal boxes with pictures of fruits. Furthermore, when [2] tested the classifier on the images of a cluttered scene, the model did okay classifying multiple objects correctly, pointing to the possibility that classifiers trained on images of individual products, may be able to identify them

among other products. However, more research is needed to explore this possibility.

We used the results in [2] as our baseline and based our process on [3]. In [3], Farren realized that the CNN model from [2]was overfitting, and attempted to see if he could take the classifier and prune or add layers and modify parameters in order to make it perform better. Thus, he ran the model on the Guided_Prunning algorithm, a greedy algorithm that improves the model accuracy step by step by taking in an array of all of the parameters of the starting model and returning a new array with parameters that would make the original model perform better. After iterating the model from [2] through the algorithm 100 times, the resulting model is less complex than the starting model with 3 CNN layers and 2 fully connected and a drop-out rate 50% on every ReLu layer. Thus, he achieved a testing accuracy 89.12% with no overfitting because of the smaller size of the model, which he concludes is better for the kinds of images in the Freiburg dataset.

Both [2] and [3] are centered around the product category recognition of individual items, whether or not they are in a group. However, within the retail product recognition space there is also work being done to give neural networks the ability to recognize objects on grocery store shelves instead of just close-ups and the ability to determine the specific brand of an object. The authors of [4] proposed a method of classifying products according to their brand by first identifying objects on shelves and then using the similarity between the objects, in their case cigarette boxes, to extract identify logos using a multi-class SVM with Gaussian radial basis function as the kernel. This achieved accuracies of about 90% for most brands. Reference [5] instead proposed a text extracting algorithm to extract text from smartphone images of grocery products, then applying a minimum distance classifier to identify characters and words. This achieved a character recognition rate of 89% and a word recognition rate of 99%.

III. METHOD

We used the Freiburg Grocery Dataset which consists of 4947 images of 25 grocery classes representing the most common categories of grocery items that exist in most homes with anywhere from 97 to 370 images per class. This dataset was collected using four different smartphone cameras at various grocery stores and homes in Freiburg, Germany, varying in the degree of lighting and clutter in order to be more representative of real-world lighting and spatial conditions. To be further representative of the state of the grocery store shelves and home pantries, for each class, the authors made sure to include a wide variety of brands and packaging design for each individual class. The images were scaled down to images of size 256x256 and padded with gray borders as needed to account for cameras with different ratios. The dataset also included a small set of 74 images of 37 cluttered scenes containing multiple classes of objects; however, for the purposes of this study, we did not consider these images.

In our study, we limited ourselves to 10 out of the 25 classes due to concerns of size, so 1,609 total images. The size of the dataset consisting of only 25 grocery classes is about 537.6 MB compared to 226.5 MB when we limit ourselves to 10 classes, which, while still big, is a lot less than the original. The 10 classes that we worked with are the following: beans, candy, fish, pasta, rice, spices, tea, tomato-sauce, vinegar, water. These are classes that we felt were not only common but also included a wide variety of possible packaging which we thought might help our model distinguish between different classes.

Using Edge Impulse, a leading development platform for machine learning on edge devices, we built several models using transfer learning and neural networks trained from scratch to evaluate their memory consumption and performance on our 10 chosen classes.

IV. EXPERIMENTS

After splitting the images into training and testing dataset using an 80/20 split, we changed the images in the dataset to grayscale and scaled them down to size 96x96 in order to lessen the memory consumption. Without pre-processing the images beforehand, the RAM usage on-device peaked at 72KB versus 4KB when we pre-processed the image and made it grayscale. It is important to note, however, even if the images were left in color, with the preprocessing, the RAM usage still peaked 4KB. After successfully preprocessing our dataset, we trained our model using two different methods on Edge Impulse with the dataset automatically split into training and testing sets, using a pre-trained neural network and training a neural network from scratch. Throughout the entire training process, we kept the learning rate at 0.0005 and the minimum confidence rating at 0.60. After training, we compared the training and testing performance on quantized variants of the models.

Transfer Learning Using MobileNetV2:

We used transfer learning from a pre-trained network in order to classify images within the 10 chosen classes from the Freiburg Grocery Dataset.

The base model was created from the MobileNetV2 model developed at Google, which was pre-trained on the ImageNet dataset, a large dataset consisting of 1.4M images and 1000 classes. MobileNetV2 is a neural network architecture that is specifically tailored for mobile and resource constrained environments [6]. The basic building block of the MobileNetV2 model is a bottleneck depth-separable convolution with residuals [6]. The model uses ReLU6 as the non-linearity because of its robustness when used with low-precision computation, kernel size 3 × 3 as is standard for modern networks, and dropout and batch normalization during training [6]. Using three different values for alpha, a value also known as the width multiplier in [6] that controls the width of the base network by proportionally decreasing the number of filters in each layer since alpha < 1.0. The alpha values tested were 0.05, 0.1, and 0.35.

MobileNetV2 0.05. The first MobileNetV2 model trained on our dataset used an alpha value of 0.05 and a final dense layer containing 10 neurons and a 0.1 dropout value. The model also had 20 training cycles, an input layer containing 9,216 features, and an output layer containing 10 features. The model had a 52.5% accuracy overall on the training data (significantly high accuracies included 76.6% for candy and 76.2% for tea) and a loss value of 1.53. The testing accuracy was 32.03% for the testing set overall.

We also estimated the on-device performance using the Edge Impulse EON compiler estimated for Cortex-M4F 64MHz (Arduino Nano 33 BLE Sense). The estimated inference time for our model was 2085 ms, peak RAM usage was 270.3K, and ROM usage was 215.2K.

MobileNetV2 0.1. The second MobileNetV2 model trained on our dataset used an alpha value of 0.1 and a final dense layer containing 10 neurons and a 0.1 dropout value. The model also had 20 training cycles, an input layer containing 9,216 features, and an output layer containing 10 features. The model had a 56.8% accuracy overall on the training data (significantly high accuracies included 79.7% for candy and 71.4% for water) and a loss value of 1.50. The testing accuracy was 42.54% overall for the testing set.

Using the Edge Impulse EON compiler, the estimated inference time for our model was 2434 ms, peak RAM usage was 270.3K, and ROM usage was 215.2K.

MobileNetV2 0.35. The final MobileNetV2 model trained on our dataset used an alpha value of 0.35 and a final dense layer containing 16 neurons and a 0.1 dropout value. The model also had 20 training cycles, an input layer containing 9,216 features, and an output layer containing 10 features. The model had a 64.3% accuracy overall on the training data (significantly high accuracies included 87.5% for candy, 78.6% for tea, and 71.4% for water) and a loss value of 1.78. The testing accuracy was 51.10% overall for the testing set.

Using the Edge Impulse EON compiler, the estimated inference time for our model was 7200 ms, peak RAM usage was 270.3K, and ROM usage was 215.2K.

Tranfer Learning Evaluations			
Pre-trained Model	Inference Time (ms)	Accuracy	Testing Accuracy
MobileNetV2 0.05	2085	52.50%	32.03%
MobileNetV2 0.1	2434	56.80%	42.54%
MobileNetV2 0.35	7200	64.30%	52.10%

Training a Neural Network from Scratch:

We intended to first train the model at the chosen baseline from [2] to determine whether that baseline would change due to the smaller size of the dataset and the classes we chose to include. This meant training the model with five convolutional layers and three fully connected layers, in line with the CaffeNet framework. From there, we would compare the performance of our baseline model with the conclusions of the [3] that the CNN in [2] was overfitting and that a simpler model works better with the types of images we are training from the Freiburg dataset.

However, we were unable to train a model of this size on Edge Impulse because the neural network was consuming too much memory and the training time exceeded the time limit set by the MLops framework. Thus, we decided to assume that the conclusions from [3] would also apply in our case since the one thing we are changing in our study is that we are only utilizing a subset of the dataset. The less complex model derived from the Guided_Prunning algorithm in [3] also proved too much for Edge Impulse as well. Therefore, from then on, we tried using the Caffe framework and [3] as a guide to come up with even simpler models architectures and test their performance.

The first attempt (**NN10**) was an extremely simple model architecture, with two 2D convolutional layers followed each by polling layers and with a dropout rate of 0.5 and 0.1 each, then 1 fully connected layer. The performance of this model was rather poor with an accuracy of 31.7% and a testing accuracy of 5.85%. The testing accuracy was determined on Edge Impulse by having the trained model classify 413 images set aside as the test set, and when classifying, the model was either 100% certain that an object was what it was, leading to a correct classification or it was uncertain because it has less than 60% confidence in the appropriate label.

There wasn't a case in which this simple model misclassified an image despite its extremely low accuracy.

Seeing this, for **the second attempt** (**NN20**), we only upped the training cycles from 10 to 20 to see if that might help, and the accuracy increased to 37.3% and the testing accuracy to 10.24%. Something notable about the testing accuracy in this attempt was that the model was extremely good at identifying vinegar within the testing set.

For **the third attempt** (**NN75**), we decided to increase the dropout rate for the second 2D convolutional layer to 0.5 and up the training cycle to 75 since any larger would result in a memory error. This resulted in a better accuracy of 45.7% and a testing accuracy of 23.17%. In this testing set, we found that the model was less uncertain than in prior attempts, with the model misclassifying some of the incorrect classifications and the correct classifications were spread across all of the 10 classes instead of being concentrated within a few classes. Vinegar, as in the second attempt, continued to have the bulk of the correct classifications.

In terms of model performance, because in the three attempts detailed above, there was no change in the number of layers in the model just some of the parameters, the on-device performance stayed constant even as the accuracy rose. The inference time was 651ms, RAM usage peaked at 126.3K, and ROM Usage stayed around 1,730.1K.

Original CNN Evaluations		
Model	Accuracy	Testing Accuracy
NN10	31.70%	5.85%
NN20	37.30%	10.24%
NN75	45.70%	23.17%

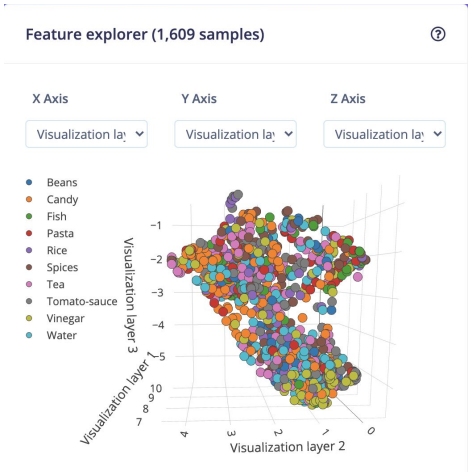
V. ANALYSIS

Overall, the MobileNetV2 model with the greatest alpha value, 0.35, produced the best accuracy for our grocery item dataset with an overall accuracy of 64.3% and a testing accuracy of 51.1%. However, one major tradeoff of using this model was that the inference time is estimated to be significantly higher than the other two attempts at using the MobileNetV2 model for transfer learning.

Even though out of all of the models we trained the MobileNetV2 0.35 model was the most accurate, we still did not manage to reach the baseline of 78.9% that we set for ourselves. Given that the MobileNetV2 is a lightweight CNN designed for mobile devices, a resource constrained device, and is intended for image classification, it makes sense, for it to have outperformed the neural networks we designed and trained from scratch. However, as for why this model was not able to reach the baseline, that could possibly be because of the dataset it was pre-trained on, which is the ImageNet dataset. The ImageNet dataset is a large dataset containing more than 1,000 categories and is not constrained to grocery store items. Therefore, it could have been that not many of the images the model was pre-trained on were similar to the images in the Freiburg Grocery dataset. Transfer Learning is most effective when the original model was pre-trained on a similar task, so perhaps the pre-training should have been on food items images instead of a general image recognition dataset.

As for the failure of the neural network trained from scratch, given more training time, training cycles and leeway to adjust with the layers and parameters without error messages coming up, we are confident that we could have achieved a respectable accuracy because even just adjusting the dropout rates without adding and getting rid of a layer caused a big increase in accuracy during training. The difficulty of training a model from scratch on a relatively small dataset of grocery store items also can be attributed to the large number of distinct products that can be identified in a grocery store and these often exceed the capability of normal object detection models [1].

The evidence of the large-scale of categories at grocery stores can be seen in the fact that the Freiburg dataset with its 25 classes, isn't even representative of all of the possible items in a store and this is without even considering the brands. In addition, the difficulty of classifying such a large set of categories is also evidenced by the feature map for 10 classes of the Freiburg dataset we worked with in this study seen below; there is no clear classification between the categories.



VI. CONCLUSION

This study is aiming to train a model using a subset of the Freiburg Grocery Dataset that beats the baseline presented in [2]. We achieved our best results of 64.3%, below the baseline, by training the dataset on a MobileNetV2 0.35 model pre-trained on the ImageNet dataset. We infer that we might have achieved greater accuracy if the model was pre-trained on a dataset limited solely to food items. We also might have achieved greater accuracy if given more training time to train a neural network model from scratch using Edge Impulse.

VII. FUTURE WORK

Given more time, we would have liked to work more on our deployment of our TinyML model on the Arduino Nano 33 BLE device since we were not able to successfully deploy our models (see the public GitHub repository for more information about this). We would love to see more research done in this space in TinyML and believe that retailers and customers alike would both benefit from implementing TinyML solutions that could range from automatic checkout, notifying store managers if an item is running out of stock, or assisting visually impaired customers in identifying items in a store.

In addition, we also would have liked to train a model that achieved an accuracy and testing accuracy well above the baseline we set for ourselves.

VIII. RESPONSIBLE AI

When thinking about the privacy considerations for our grocery item detection mechanism and deploying responsible AI in the real-world, it is important to think about the context, key actors, attributes of the data, and transmission constraints on the flow of information. Our application's context is a grocery store, the key actors are the grocery store employees, owners, managers, and customers of the store, the attributes of the data are images taken by cameras installed throughout the store, and the transmission principles are that the store would control how image detection data is collected but the system would only output information regarding food items that are recognized and trained through the system.

Due to the potential of capturing human images in our grocery item detection deployment, there is a slight chance that privacy may be violated by capturing people accidentally and without their knowledge. However, we believe that this application would still ultimately be desirable because of its benefits to the retail and consumer space.

Our application in improving and adapting grocery image detection for the TinyML space would support the contextual values for grocery stores in giving people access to what they need quickly. By using TinyML in retail, stores would be able to provide their customers with better access to goods and allow customers to automatically checkout on their own.

In regards to if this application would promote autonomy, grocery item detection certainly allows customers to shop according to their own considerations and desires and should improve their shopping experience overall. Additionally, our application used for an elderly or visually impaired population who may have trouble locating items or reading labels may benefit from a similar application built based on their specific needs.

On the topic of power relations, the power dynamic in a grocery store being between the store owners and managers and customers, we do not believe that our application would contribute significantly to power relations. Our application would not store customer data to target specific customers' purchasing habits. Therefore, it would neither improve or harm power relations.

Finally, our application offers a fair distribution of burdens and benefits. For benefits to the stores and customers, while the stores can hire less employees and save money, customers could benefit from an enhanced shopping experience in a store using an automatic check-out method and similar applications for grocery item detection.

REFERENCES

[1] Y. Wei, S. Tran, S. Xu, B. Kang, and M. Springer, "Deep Learning for Retail Product Recognition: Challenges and Techniques," *Comput. Intell. Neurosci.*, vol. 2020, pp. 8875910–8875910, 2020, doi: 10.1155/2020/8875910.

[2] P. Jund, N. Abdo, A. Eitel, and W. Burgard, "The Freiburg Groceries Dataset," 2016, Accessed: Dec. 16, 2020. [Online]. Available: https://arxiv.org/abs/1611.05799.

[3] D. Farren, "Classifying food items by image using Convolutional Neural Networks," *Unpublished*, p. 6.

[4] G. Varol and R. S. Kuzu, "Toward retail product recognition on grocery shelves," 2015, vol. 9443, pp. 944309–944309–7, doi: 10.1117/12.2179127.

[5] F. Einsele and H. Foroosh, "Recognition of Grocery Products in Images Captured by

Cellular Phones," *Int. J. Comput. Inf. Eng.*, vol. 9, no. 1, pp. 159–163, Jan. 2015.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

APPENDICES

APPENDIX A: System Description

We included a clear description of how to deploy the system in the README.md file within the GitHub Repository: https://github.com/jessie9111/grocery-recognition-tinyml. The repository contains the sketch used to test out the ArduCam 5MP Camera interfacing with the Arduino Nano 33 BLE, the final TFLite files and descriptions for each file, instructions on how to run each model using our sketch, and descriptions on the Edge Impulse settings we used to train the models discussed in this paper. We were not able to successfully deploy our models on the Arduino Nano 33 BLE board by the time this paper was submitted, so the repository describes the errors that occur when we try to deploy the model.

APPENDIX B: Group Contribution

Jessica Edwards: Jessica worked on the code used for the final project to deploy the model to the Arduino and documented the code. She also worked on building the transfer learning models and helped frame the Introduction, Method, Experiments, Analysis, Conclusion, Future Work, and Responsible AI sections of the final report.

Fatima Mbaye: Fatima researched related work regarding image classification and object recognition in the retail space, as well as successful applications of the dataset and algorithms we wanted to use and base our models off of. She built the Neural Network models and contributed to the Related Work, Method, Experiments, Analysis, and Conclusion section.