Issues in Information Systems
Volume 26, Issue 2, pp. 43-57, 2025

DOI: https://doi.org/10.48009/2_iis_104

Mobile app for identifying recyclable items using a convolutional neural network

George Stefanek, *Purdue University Northwest, stefanek@purdue.edu*
Kody Smart, *Purdue University Northwest, smartk@purdue.edu*
Mark Kadah, *Purdue University Northwest, mkadah@pnw.edu*
Eric Shelton, *Purdue University Northwest, sheltoe@purdue.edu*
Nathan Custin, *Purdue University Northwest, ncustin@purdue.edu*

Abstract

This project focused on developing a mobile app that could identify recyclable items and provide instructions for recycling or disposal. The use of such an app can help in appropriately recycling items, increasing the number of items that a household might recycle and improving the effectiveness and speed of recycling at the recycling plant. To easily identify items, the app uses the phone's camera to allow a person to take a picture of an item. The first phase of the project created a mobile app that sends a photo to a cloud server running a Convolutional Neural Network (CNN) that performs the classification. The server application then looks up recycling instructions on a MySql database and sends back the instructions to the mobile app. The mobile app also has a local version of the database so that items can be looked up locally. A key focus of the project explored the public datasets that were available for recycling, trained various mobile-oriented CNNs to determine their effectiveness using these public datasets, tuned hyperparameters for optimum performance of each model, and selected the best CNN, dataset and framework that could ultimately allow the app to be deployed natively on a mobile platform.

Keywords: neural network, CNN, machine learning, recycling, mobile app

Introduction

Recycling garbage is done routinely in many households; however, recycling is typically underutilized. A report on present and future residential recycling in the U.S. (Appel, M. et al., 2024) states that 21% of residential recyclables are being recycled even though 43% of households participate in recycling. The study also finds that of those households that participate only 57% of recyclable material is put in recycling containers which means that households are not fully participating. Some items such as cans, bottles, glass items, and paper cartons are more obvious as recyclable items that can be appropriately put into a recycling container and picked up by a garbage service. Other items particularly various plastics, rubber items, metals, items that contained food, woven materials, paints, batteries, machines, televisions, computers, and other electronic items may not be recyclable or have special instructions for recycling or disposal. When it is unclear of how to dispose of an item, households may end up throwing items out with regular garbage. Disposal and recycling instructions are typically posted on the recycling service website which require an individual to use a computer or phone, navigate through the web pages and search for instructions which

makes it cumbersome and unlikely to be used when trying to quickly dispose of an item. This emphasizes the need for better ways of communicating this information.

To improve the quantity of items that get recycled and encourage people to recycle, one approach is to use technology to make it easier and quicker for households to determine what to do with an item. Inspiration can be taken from existing recycling mobile apps such EcoScan, Scrapp, and GreenScanr that are used to help people recycle items. Other mobile apps that use cameras to identify plants, birds, animals, bird calls, etc. can also help in proposing a design for a recycling app that uses the camera to identify items. Typically, with these types of apps the user opens the app and uses a phone's camera to identify something of interest. Some examples include "Seek" by iNaturalist (Seek by iNaturalist, n.d.), "PlantSnap" by PlantSnap (PlantSnap, n.d.), and Merlin Bird ID (Merlin Bird ID, n.d.). Additionally, a machine learning model that can detect recyclables of various shapes and states may be able to be applied in the recycling facility itself.

Problem Statement

The limitations that these apps may have are the ability to identify an item correctly when it is broken, crumpled, obscured, dirty, in various orientations, and has a background or light condition that make it more difficult to correctly identify. There exists a need for more robust data sets to overcome these issues. Some of the existing apps require the models to run on a cloud server. Small models may be able to be run natively on a mobile app, but larger models that have been trained with large datasets that include images with the limitations mentioned above may perform better but may have to be hosted on the cloud to have the processing power that may be required. Creating a mobile model that can have near real-time performance natively on a mobile app yet provide high accurate classification rates of items in various states would require experimentation with hyperparameters of the model and a dataset that is well balanced at representing these items, but not overly large.

Literature Review

There are a few mobile apps that focus on recycling. There are two mobile apps called EcoScan. The first EcoScan app uses an AI model developed with PyTorch with over 15,000 images to provide recognition of recyclables with 94% accuracy and then directs users to nearby recycling locations using Google Maps and provides tips for recycling (Scan, Recycle, Transform with EcoScan, n.d.). It also uses barcodes to identify items. The second ecoScan app also takes a photo or scans the barcodes of an item and claims an accuracy of 98% (EcoScan, n.d.). These apps also build-in gamification to try to help the user do more recycling. The GreenScanr app (GreenScanr, (n.d.)) uses an AI model or barcodes to identify images of recyclable items and provide information on material composition, weight, and carbon avoidance. The Scrapp app (8 out of 10 of us recycle incorrectly, (n.d.)) uses barcodes or manual search to get recycling instructions, discover drop-off points and learn about zero-waste habits.

Other non-recycling apps that also use a mobile phone camera to identify things include the "Seek" app by iNaturalist (iNaturalist, (n.d.)) which identifies wildlife, plants, and fungi by using the camera. The Seek app uses computer vision algorithms to analyze the image and suggest possible identifications based on the iNaturalist database (Iwane, T., 2019). The app used the TensorFlow deep learning framework with Nvidia hardware to train the neural network on the iNaturalist database of images. These images have been labeled by the site's community of experts with approximately 4,000,000 verifiable observations vetted by experts and represent 100,000 species (Gee, S., 2017). iNaturalist determined that having at least 20 research grade observations was necessary to include a species in its model.

The Plantsnap mobile app (World Leading Plant Identification Technology, n.d.) uses AI and machine learning to identify nearly any plant, using only a smartphone photo. When you open the app, it immediately takes you to a camera view. Plantsnap uses the machine learning services of Imagga, a company that provides cloud-based visual AI services using an API. The model is trained on 320,000 plants which used a training set of 90 million images (Smith, T., 2019).

Merlin Bird ID developed by Cornell Lab of Ornithology uses machine learning to train models of bird vocalizations and photos (Merlin Bird ID, n.d.). Merlin converts the audio into an image called a spectrogram. The spectrogram plots sound frequencies that appear in the recording, as a function of time. This spectrogram image is then fed into a modern computer vision CNN model. This model was trained to identify birds based on 140 hours of audio containing bird sounds, in addition to 126 hours of audio containing non-bird background sounds, like whistling and car noises. For each audio clip, a group of sound ID experts from the Macaulay Library and the eBird community found the precise moments when birds were making sounds and tagged those sounds with the corresponding bird species.

Most of the apps above used convolutional neural networks (CNN) for creating a machine learning model trained on actual images of interest. The use of CNNs for image classification is supported by Lai (2019) who compared the use of a SVM (Support Vector Machine) which is a traditional classifier with a CNN which is a deep learning model. The SVM algorithm distributes learning samples into two different classes in a high-dimensional space by linear classification.

The CNN was constructed as a three-layer model with layer 1 having 32 3*3 convolution kernels to collect the feature information of the image. The image data was from the Mnist dataset which contained handwritten digits. The digits have been size-normalized and centered in a fixed-size image (28 X 28). Of the 70,000 images in the training set, 60,000 images were used for training and 10, 000 images for testing. They found that the CNN performed better than the SVM achieving an image recognition rate of 98.85% vs 93.92%.

Pak and Kim (2017) specifically reviewed different deep learning models including CNNs on how well these models classified images. They looked at the results from several of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) contents which are the largest object recognition contests held in recent years. The contests used an image database of 15,000 images that can be downloaded from Kaggle. They evaluated AlexNet, VGG, GoogLeNet, and ResNet. The AlexNet machine learning model consisted of a network, which had 60 million parameters, eight layers, and had an error rate of 16.4% in the ILSVRC-2012 contest.

The VGG machine learning model consists of 16 weight layers including thirteen convolutional layers with a filter size of 3x3 and three fully connected layers and had an error rate of 7.3% which took second place in ILSVRC-2014. The GoogLeNet model had 22 layers when counting only layers with parameters and won the ILSVRC-2014 challenge with 6.7% error. The ResNet CNN network that was used consisted of 152 layers and won first place in ILSVRC-2015 with an error rate of 3.6%.

Additionally, Li (2022) reviewed literature to see how effective various deep learning techniques were to medical image recognition. It was found that a CNN was used to identify and classify cervical examination images with 89.1% accuracy on the test set. A CNN model was also used to classify breast density to determine the risk of breast cancer with 98.8% effectiveness. Also, histopathological images of lung cells were trained using a CNN model which was able to accurately distinguish diseased cells from normal cells with an accuracy of 97.0%.

Research Questions

The use of a CNN for classifying images has been shown to have success in various applications and was a good candidate for a deep learning model to be used in detecting recyclable waste. The research questions in the development of this recycling app were: 1) identify image datasets that currently exist for recyclable items, 2) train a CNN on these datasets and determine how well these datasets recognize recyclable items and experiment with hyperparameter configuration to achieve better classification, 3) determine how well this initial model classifies recyclable items that have some of the issues as stated in the problem statement above, 4) determine what machine learning platforms that will support running the machine learning model on a mobile device, 5) select a cloud service that can host the machine learning component, and 6) determine if the initial dataset should be augmented with additional training examples to increase accuracy of identification of items in various states.

Methodology

The methodology was to: 1) identify the technologies that can to be used for building the mobile app and run a machine learning model natively on a mobile app, 2) choose the machine learning model, 3) select a cloud environment and supporting technologies to also host the machine learning component, 4) define the UI requirements for the mobile app, 5) find and select the initial datasets that could be used for recycling, 6) design the architecture for the app, 7) build an initial prototype app that uses the initial datasets and host the machine learning and database components on the cloud to show a proof of concept, and 8) test how well the app can classify various categories of recyclable items using pictures that do not include some of the problems as identified in the problem statement. A future version of the mobile app will be created to run the machine learning and database components natively on the mobile device. The standalone version may need to be constrained by the machine learning model and size of the dataset on which the model is trained for it to run responsively on a phone or tablet. Future work will augment the dataset with additional pictures that include those as identified in the problem statement.

Mobile App Development Technologies

To reduce the amount of time it takes to develop apps that run on multiple operating systems such as Android, iOS and Windows, a cross-platform development environment was selected. Out of the multiple cross-platform development environments that are currently available such as Flutter, React, and MAUI, the Microsoft MAUI cross-platform platform was selected. The MAUI platform was selected since it uses C#, a full-featured language, as the underlying language to handle event processing and programming of controls, uses XAML that can be used for building the front-end, and has access to many libraries that can be used with C#. Alternatively, a Blazer version of MAUI can be used to develop an app where the user-interface is built on top of HTML/CSS.

Additionally, Flask (Flask, 2010) was used to simplify communication to the machine learning application that was initially hosted on the cloud. Flask is a lightweight web application framework written in Python that does not require particular tools or libraries and is based on the Werkzeg Web Server Gateway Interface (WSGI) toolkit and the Jinja2 template engine. WSGI is the specification of a common interface between web servers and web applications. Also, with 'webview' technology, a Flask web app can run in a native container on iOS and Android. To run a machine learning model on a mobile platform, TensorFlow Lite was investigated (Shea, M., n.d.). TensorFlow Lite (now LiteRT) is a platform to run machine learning models on mobile devices. All the workflow is executed within the mobile device, which avoids the need to send data back and forth from a server hosting the model. A typical workflow using TensorFlow Lite consists of creating and training a machine learning model in Python using TensorFlow, converting the

trained model in a suitable format for TensorFlow Lite using TensorFlow Lite converter, and deploying the model on the mobile device using TensorFlow Lite interpreter (Ippolito, P., 2019). The MySql database was used on the cloud to hold the recyclable item description, information and disposal instructions. The SQLite database was used locally on the app.

Datasets

A dataset was found on GitHub called "Trashnet". The dataset was already broken down into categories with the images labeled and resized to 224x224. Trashnet was a good start to the project but was very limited to only 2,527 images. After training a model with this dataset, the model could attain accuracy in the mid-eighties but struggled to consistently classify the test images. Thus, the search for a larger dataset began. Another recycling dataset was found on Kaggle called "Recyclable and Household Waste Classification" which expanded the training set from 2,527 images to 15,000 and categories from 10 to 30.

Machine Learning Model and Training

The machine learning model that was selected was a CNN supported by the literature search as the most effective model for image classification. The specific CNN models that were selected for testing were ResNet50, MobileNetV2k, and for increased efficiency EfficientNetB0 and InceptionV3 models. ResNet CNNs were introduced in 2015 with the 50-model having 50 layers making it considerably deeper than earlier CNNs and the ResNet name coming from the use of residual blocks that skip connections to bypass certain layers make it more efficient in training (Residual Network (ResNet) – Deep Learning, 2025). EfficientNet-B0 is a CNN that is known for its efficiency and good performance in image classification tasks. It is trained on over a million images from the ImageNet database and uses a scaling method to uniformly scale the network's depth, width, and resolution as opposed to other CNNs that arbitrarily scale individual dimensions which leads to a better balance between model size, computational cost and accuracy (Efficientnet Architecutre, 2024). InceptionV3 is a CNN that uses factorized convolutions, label smoothing and batch normalization to enhance efficiency and accuracy (Inception V2 and V3 – Inception Network Versions, 2022). The original intent of the app was to run the model natively on the mobile device, so the MobileNetV2 model was added for evaluation. This is a CNN architecture that was designed by Google for use on resource limited devices such as cell phones and other personal electronic devices.

The PyTorch framework was chosen for its suite of preprocessing tools and data loaders. The initial work with PyTorch was on using the base model to perform the inference, but the results were lackluster. Therefore, other base models were tested for feature extraction with the goal of building a custom implementation of a pretrained model. The following models were evaluated: 1) ResNet50, 2) MobileNetV2, 3) EfficientNetB0, and 4) InceptionV3. All the models finished with nearly the same results, with the only differences being attributed to time complexity. The EfficentNetB0 and InceptionV3 models were not as accurate on the original dataset as MobileNetV2. Since the intent was to perform inferences natively on the mobile device, the lighter weight MobileNetV2 was chosen. It is important to note that MobiileNetV2 has the advantage of having low compute requirements in the interest of speed and efficiency, but it does sacrifice some accuracy to accomplish those goals. The "heavier" models should be considered if the model would get trained with a much larger dataset with more categories of items. Google has made some improvements upon MobileNetV2 with V3, but the model architecture was not explored during this prototype phase.

For optimum performance tuning the hyperparameters was done next. There is a hyperparameter tuning framework available called Optuna. With the use of Optuna, the hyperparameters that were focused on were: 1) learning rate, 2) optimizer, 3) weight decay, 4) batch size, 5) dropout rate, and 6) scheduler type. The model proved to be most sensitive to changes in the learning rate and the optimizer used during the training process. It was found that AdamW and SGD were the two preferred optimizers, both without a

scheduler being used. Additionally, hyperparameter tuning with Optuna could provide for further increased accuracy and precision.

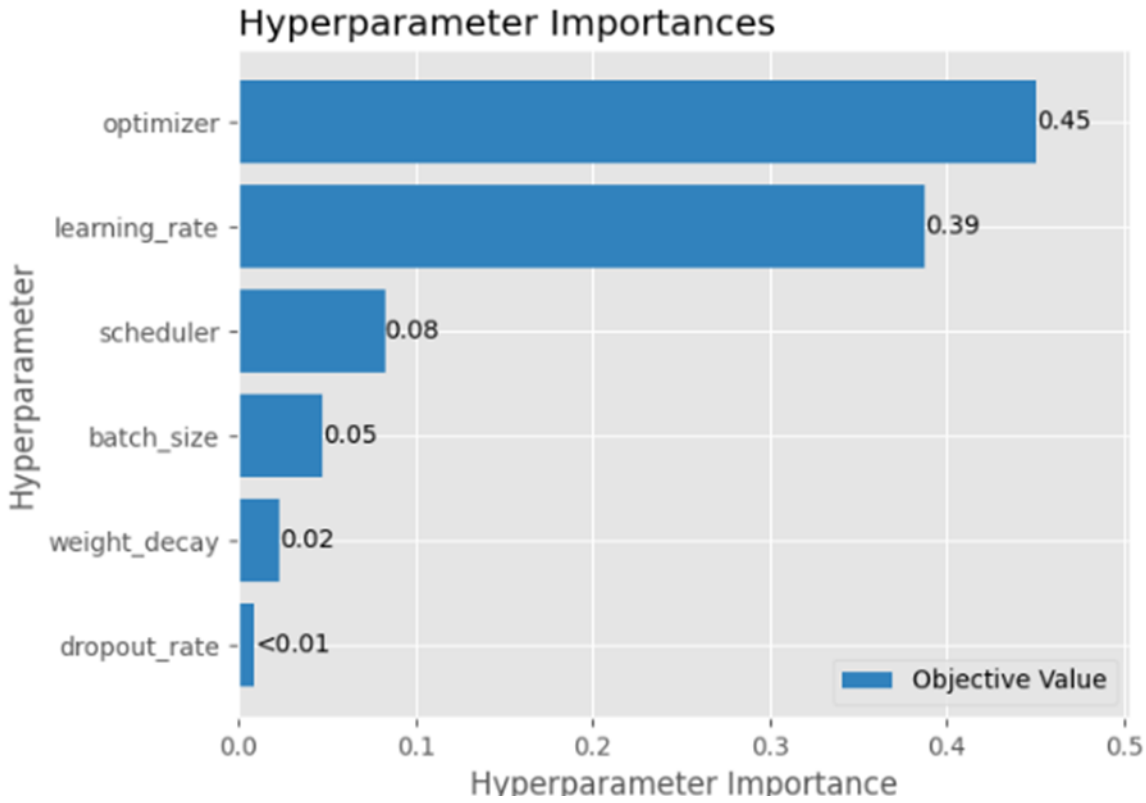

Figure 1. Hyperparameter Impact

The training used MobileNetV2 as the base model with its trained layers set to a frozen, untrainable state. Using the model in this way allowed the use of a pretrained model for feature extraction for improved recognition ability on the new data. Once the training reached a plateau, the pretrained model was then unfrozen. The last ten layers of MobileNetV2 were then trained on the new dataset increasing the model's accuracy from 85% to ~92%.

Requirements for the User-Interface of the Recycling Mobile App

A local engineering company in collaboration with a local recycler had requested the development of this custom app that could be used by the local population to help in finding instructions on how to recycle various items. Also, there was interest in how the identification of items using machine learning could be used directly in a recycling plant. The requirements of how the initial prototype app user-interface would work are described in Figures 2 and 3.

When a user has an item that they would like to recycle and are not sure of the steps they need to take, they could use the app to find information and instructions for recycling and disposal of the item. To do this, the user would open the application on their mobile device and would be prompted with two options: "Take a Photo" and "Search". If the user decides that they would like to take a photo of their item, then they could choose that option and their device's camera would be engaged. The user could then take as clear a photo as possible which would be sent to a cloud server for classification. The image would be processed by the machine learning model (MLM) hosted on the cloud server to attempt to classify the item of interest. The

classified item's label would then be searched for a match in a database of recyclable items that includes instructions for disposition. If a match is found in the database, the information and instructions would be sent back to the app and displayed to the user in a simple "Item's Details" page.

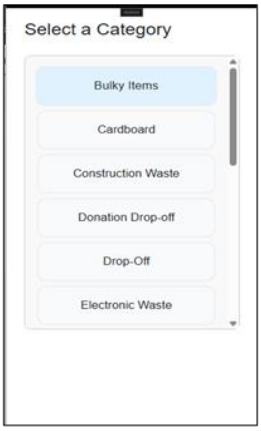

Figure 2. Manual Selection of Recyclable Items by Category in UI

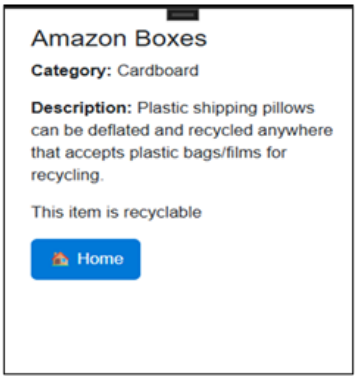

Figure 3. Instructions for Disposal of a Recyclable Item Displayed in UI

If the user prefers to manually search for an item, then they could navigate to a search page by clicking the "Search" button, which would open another page that includes drop-down menus containing all the items in the database. The user can navigate through the menus to find an item with the corresponding recycling and/or disposal instructions which would be looked up in the local SQLite database.

Mobile App Architecture

The app was designed to run on a user's mobile device running either the Android, iOS or Windows operating system and is shown in Figure 4.

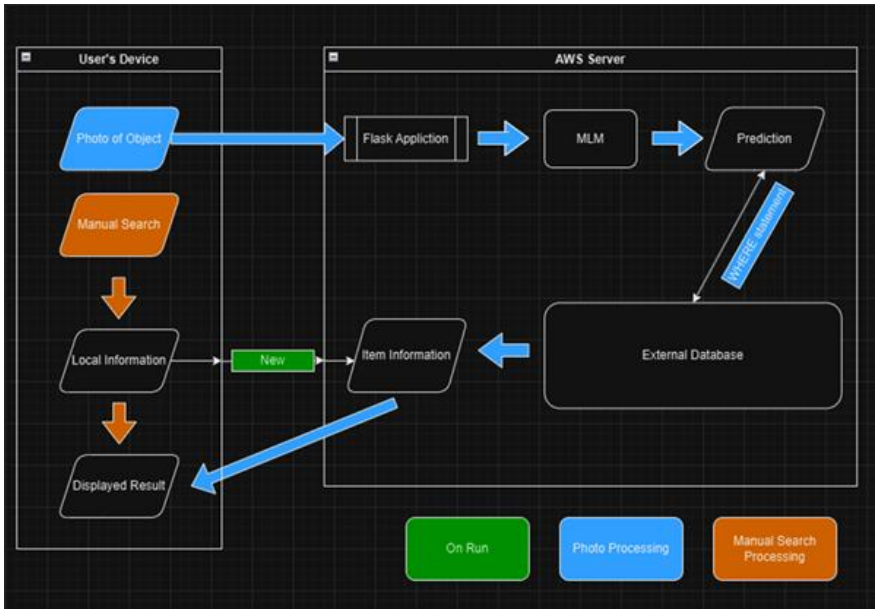

Figure 4. App Architecture

When a user takes a photo, it is sent as a file stream up to a Flask application on the AWS server where the image is processed by the trained machine learning model. Once a classification is made, the item's label will be returned as a string and used in a SQL WHERE statement to query the MySQL database also hosted on AWS for the item's recycling information. That information is then sent back to the mobile app and displayed on the user's device. If the user decides they would like to manually search for an item, then they can open the search page in the app and search for it. Once an item is selected, the item label will be used in a SQL query WHERE statement to search for it in the local SQLite database that contains all the data synched from the cloud database.

The flask application that facilitates running the machine learning model to classify an image sent from the mobile app was hosted on the AWS server. The technologies used were Flask, Torch, Torchvision and Pillow. The following pseudocode in Figures 5 and 6 describes setting up Flask and the Python code to run the model on the cloud server. The mobile application uses the HttpClient class to send requests to the Flask API and retrieve results from the classification which is described in the pseudocode in Figure 7. The mobile app code is implemented in a Razor file in the Maui Blazer implementation of the mobile app.

```
START

IMPORT necessary libraries and modules:
    - Flask for web server
    - request and jsonify for handling requests and JSON responses
    - PIL for image handling
    - torch and torchvision for model operations
    - JSON for reading class names
    - Custom model definition from model_def.py

INITIALIZE Flask app

DETECT if CUDA (GPU) is available, else use CPU

LOAD class names from JSON file

INITIALIZE the model with number of classes

LOAD trained model weights

MOVE model to appropriate device (CPU or GPU)

SET model to evaluation mode

DEFINE image preprocessing steps:
    - Resize image to 224x224
    - Convert image to Tensor
    - Normalize image with standard ImageNet means and stds

DEFINE route `/predict` with POST method:
    IF no file uploaded:
        RETURN error JSON response with status code 400

    OPEN the uploaded image and convert to RGB
    APPLY preprocessing steps to image
    ADD batch dimension and move to device

    PERFORM model inference without gradient tracking
    GET predicted class index by finding class with highest score
    MAP class index to corresponding label using class names

    RETURN JSON response with prediction label

DEFINE route `/` with GET method:
    RETURN JSON indicating API is running

IF this file is run directly:
    RUN the Flask app

END
```

Figure 5: Pseudocode of Flask Setup Logic

```
from flask import Flask, request, jsonify
import pickle
import numpy as np

app = Flask(__name__)
# Load the pre-trained model
model = pickle.load(open("model.pkl", "rb"))
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    features = np.array(data["features"]).reshape(1, -1)
    prediction = model.predict(features)
    return jsonify({'prediction': prediction.tolist()})
if __name__ == '__main__':
    app.run(debug=True)

# This Flask app loads a trained model saved as model.pkl.
# It listens for POST requests at the predict endpoint.
# When a request with input features arrives, it passes the data to the model for prediction.
# The predicted output is returned in JSON format.
# Flask and any dependencies must be installed prior to running this program:
# pip install flask numpy pickle-mixin
```

Figure 6. Python Code that runs Model using Flask.

```
DEFINE class ImageUploader
    DEFINE static HttpClient instance
    DEFINE asynchronous method SendImageAsync with parameter imagePath

        SET url to the API endpoint for image prediction

        CREATE a new MultipartFormDataContent to hold the image file

        OPEN a file stream to read the image from imagePath

        CREATE a StreamContent object using the file stream

        DETERMINE the image's MIME type:
            IF image file extension is .png
                SET mimeType to "image/png"
            ELSE
                SET mimeType to "image/jpeg"

        SET the Content-Type header of the stream to mimeType

        ADD the stream content to the multipart form data with:
            - field name as "file"
            - file name as the image file's name

        SEND a POST request asynchronously to the API url with the multipart form data

        IF response status is unsuccessful
            THROW an exception

        READ the response content as a string asynchronously

        RETURN the response string

    END METHOD

END CLASS
```

Figure 7. Pseudocode for Method in MAUI Blazer App to Communicate with Cloud Flask App

Results

The trained CNN MobileNetV2 model had training accuracy in the mid-nineties and validation accuracy in the lower to mid-eighties. The model was suffering significantly from overfitting with the small and invariant dataset. Once the model reached the point where it hit a plateau, the testing phase began. Initial testing was performed on random images pulled from a Google images search.

The model showed some promise at classifying recyclable images with a precision score of 92% across all 30 categories of items. Although the accuracy numbers are not as high as we would like to get them, the model is able to perform relatively well, misclassifying four of the 30 test images. Confusion matrices for some of the recyclable categories are shown in Figures 8 – 11 and Table 12 shows the training statistics for a subset of items.

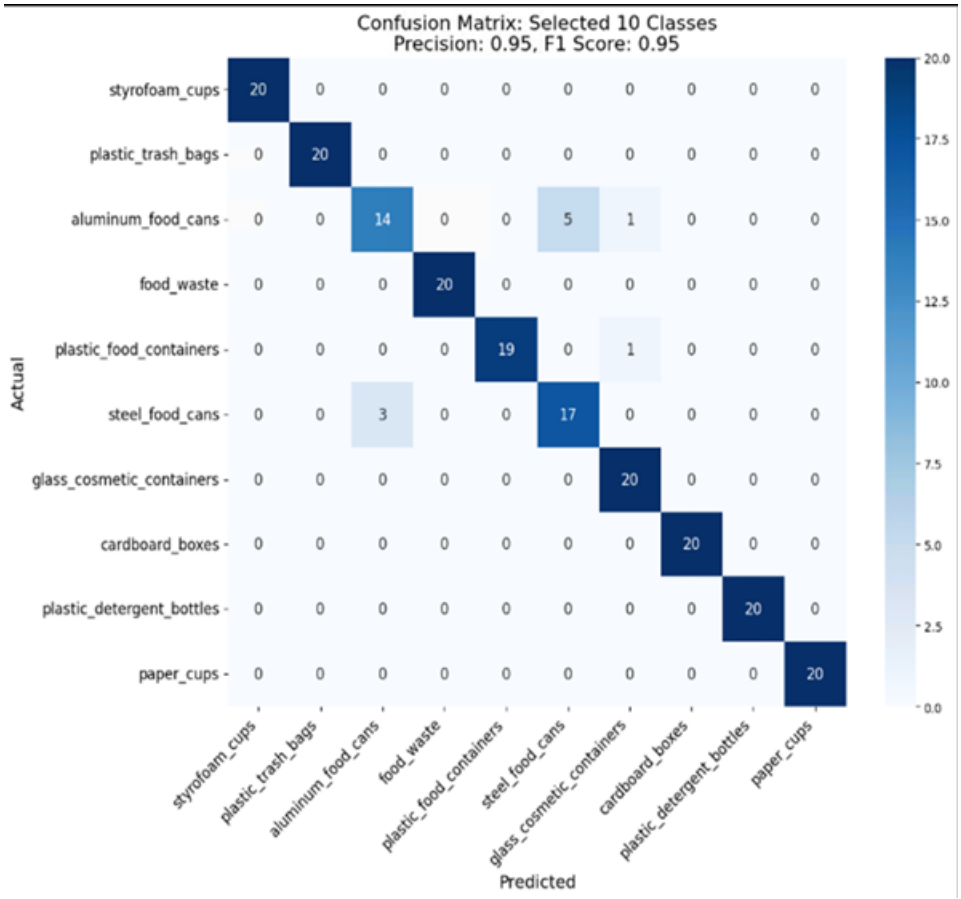

Figure 8. Confusion Matrix for First Set of 10 Categories of Recyclables

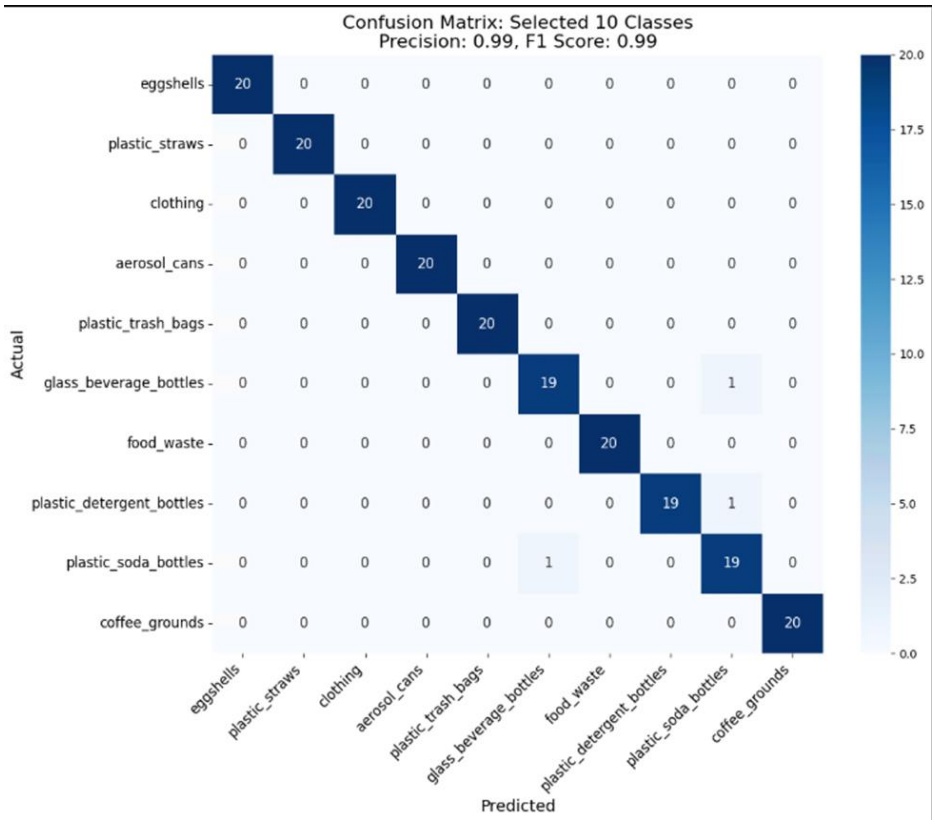

Figure 9: Confusion Matrix for Second Set of 10 Categories of Recyclables

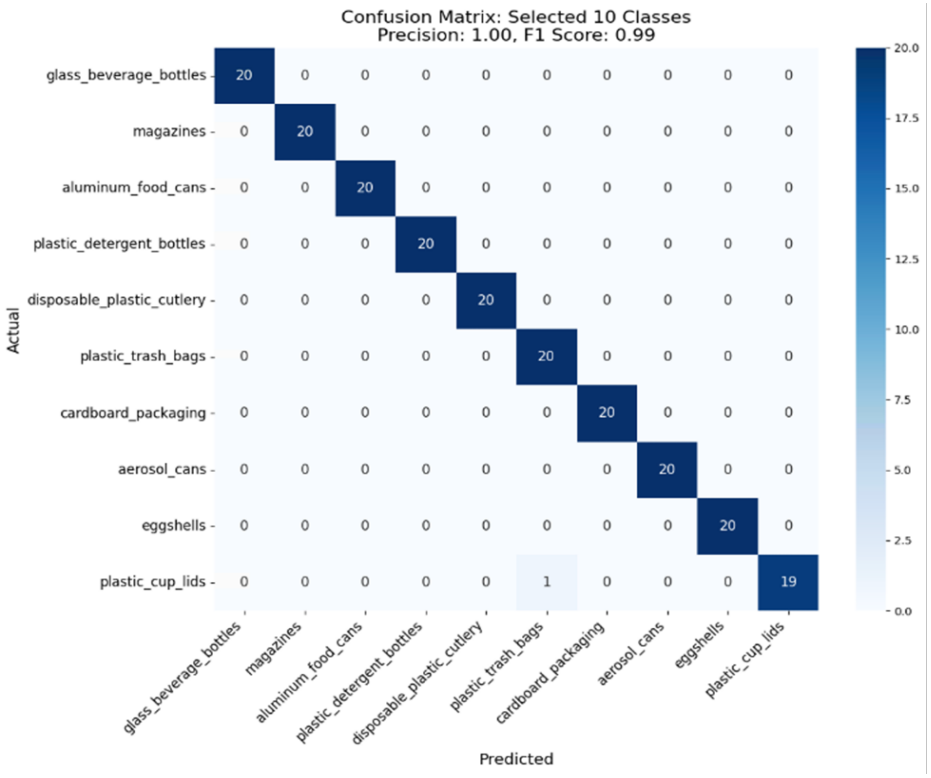

Figure 10: Confusion Matrix for Third Set of 10 Categories of Recyclables

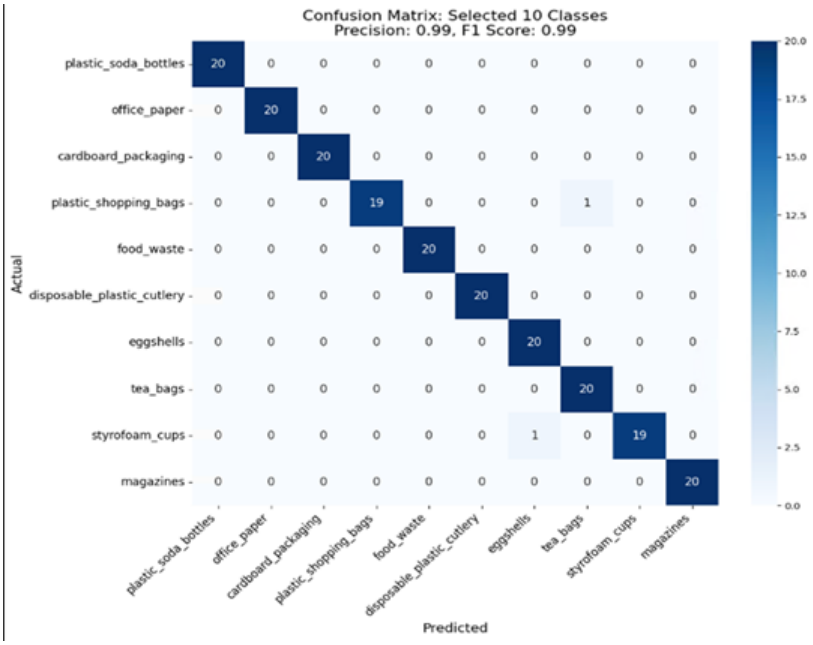

Figure 11. Confusion Matrix for Fourth Set of 10 Categories of Recyclables

Table 12. Recyclable Item Training Statistics for Subset of Common Items

Recyclable Item	precision	recall	f1-score
aerosol_cans	0.977911647	0.970119522	0.974
aluminum_soda_cans	0.891181989	0.95	0.9196515
cardboard_boxes	0.71192053	0.86	0.778985507
clothing	0.967676768	0.958	0.96281407
glass_beverage_bottles	0.971428571	0.952	0.961616162
glass_food_jars	0.943074004	0.994	0.967867575
magazines	0.995850622	0.96	0.977596741
newspaper	0.918714556	0.972	0.944606414
office_paper	0.948979592	0.93	0.939393939
plastic_detergent_bottles	0.986027944	0.988	0.987012987
plastic_food_containers	0.961770624	0.956	0.95887663
plastic_soda_bottles	0.941414141	0.932	0.936683417
plastic_trash_bags	0.981744422	0.968	0.974823766
plastic_water_bottles	0.947261663	0.934	0.940584089
steel_food_cans	0.754221388	0.804	0.778315586
styrofoam_food_containers	0.960784314	0.98	0.97029703

Discussion

The training set used was the “Recyclable and Household Waste Classification” training set from Kaggle. The results showed that on average there was a classification accuracy of approximately 92% across all

categories using the MobileNetV2 model and the training set. The pictures sent to the model from the mobile phone camera were intact items in good lighting. The results of the prediction on the test images showed that the model had some difficulty when classifying different versions of the same item. For example, the dataset contains images for both aluminum food cans and steel food cans. It would be a challenge for a human to make the distinction between the two from an image. This dataset had good images that were not broken, crumpled, obscured, dirty, or in various orientations. This may be adequate for identifying many recyclable items at home but would not do well in suboptimal conditions or with items that are not perfectly intact. The dataset with suboptimal images would need to be compiled in a future phase of the project. Pictures of damaged and broken items were not identified well, and it was clear that the dataset would need to be augmented with more suboptimal pictures of these items.

Conclusions

It is concluded that the first version of the app performed well enough so that it could be ported to a mobile platform in the next iteration of this project. For most recyclable applications it would work relatively well. Additional configuration using hyperparameter setting may slightly improve performance. However, additional work will need to be done to compile a more versatile dataset with images that show items that are broken in various orientations, obscured as identified in the problem statement. *Public datasets with broken, crumpled, and obscured recyclable items would be explored and augmented with additional photographs taken by the developers. Image processing software may be used to automate the creation of images under various orientations, light conditions and fuzziness.* The model created using the "Recyclable and Household Wase Classification" dataset would be further fine-tuned with these additional non-optimal images of recyclable items. It may be a challenge finding such images already compiled in a public repository so it may require compiling at least some of the sub-optimal recyclable images by taking photographs of the various items and creating our own dataset. This dataset would be relatively small but could enhance the classification of items and yet run on the mobile platform.

To get near an ideal classification may require a much larger dataset which when used in fine-tuning the model may have to be run on the cloud where faster resources are available under the infrastructure designed in this initial project. The model could be also enhanced by the app asking the user to identify the recyclable category for the item and adding the image with label to the database for use in future training of the model. Additional fine-tuning and training of the model for use on a mobile app would require experimentation to create a dataset and model that can run with high accuracy and minimum processing power. Finally, the UI will be further refined for better look feel and tested for usability and user experience by a cohort of users.

References

8 out of 10 of us recycle incorrectly. (n.d.). Scrapp. https://www.scrappzero.com/products/mobile-app

Appel, M, Francis, A., Payne, A., Tanimoto, A & Mouw, S. (2024). *"State of Recycling: The Present and Future of Residential Recycling in the U.S. | 2024"*. Retrieved on 05/14/2025 from https://recyclingpartnership.org/wp-content/uploads/dlm_uploads/2024/05/SORR_Methodology-1-1.pdf

Efficientnet Architecture. (2024, Jun 3). GeeksforGeeks. https://www.geeksforgeeks.org/efficientnet-architecture/

Flask. (2010). Pallets. https://flask.palletsprojects.com/en/stable/

How to Convert a Flask App to Mobile Apps for iOS and Android. (2025). Mobiloud. https://www.mobiloud.com/use-cases/flask#:~:text=With%20%27webview%27%20technology%2C%20your,running%20as%20a%20mobile%20app.

Inception V2 and V3 – Inception Network Versions. (2022, Oct 14). GeeksforGeeks. https://www.geeksforgeeks.org/inception-v2-and-v3-inception-network-versions/

Gee, S. (2017, June 18). *iNaturalist Launches Deep Learning-Based Identification App*. https://www.i-programmer.info/news/105-artificial-intelligence/10848-inaturalist.html/

GreenScanr. (n.d.). DC1. https://www.datacompanyone.com/greenscanr-app

Hoffman, B. & Van Hom, G. (2021, Jun 22). *Behind the Scenes of Sound ID in Merlin*. CornellLab Macaulay Library. https://www.macaulaylibrary.org/2021/06/22/behind-the-scenes-of-sound-id-in-merlin/

iNaturalist Computer Vision Explorations. (n.d.). iNaturalist. https://www.inaturalist.org/pages/computer_vision_demo

Ippolito, P. (2019). *How to Deploy Machine Learning Models on Mobile and Embedded Devices.* https://www.freecodecamp.org/news/machine-learning-for-mobile-and-embedded-devices/

Iwane, T. (2019, April 19). *Real-time Computer Vision predictions in Seek by iNaturalist version 2.0.* iNaturalist. https://www.inaturalist.org/blog/23075-real-time-computer-vision-predictions-in-seek-by-inaturalist-version-2-0

Lai, Y. (2019, October). A comparison of traditional machine learning and deep learning in image recognition. In *Journal of Physics: Conference Series* (Vol. 1314, No. 1, p. 012148). IOP Publishing.

Li, Y. (2022, January). Research and application of deep learning in image recognition. In *2022 IEEE 2nd international conference on power, electronics and computer applications (ICPECA)* (pp. 994-999). IEEE.*Merlin Bird ID. (n.d.). CornellLab. https://merlin.allaboutbirds.org/*

Model optimization. (2024). LiteRT. https://www.tensorflow.org/lite/performance/model_optimization

Pak, M., & Kim, S. (2017, August). A review of deep learning in image recognition. In *2017 4th international conference on computer applications and information processing technology (CAIPT)* (pp. 1-3). IEEE.

Recycle waste. (n.d.). EcoScan. https://www.ecoscanapp.eu/

Residual Networks (ResNet) – Deep Learning. (2015, Apr 7). https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

Scan, Recycle, Transform with EcoScan. (n.d.). EcoScan. https://www.ecoscan.tech/

Seek by iNaturalist. (n.d.). iNaturalist. https://www.inaturalist.org/pages/seek_app

Shea, M. (n.d.). *TensorFlow Lite Inception Model Android Tutorial*. https://www.youtube.com/watch?v=8zQsAl2z4iU

Smith, T. (2019, Sep 22). *Plantsnap and Imagga Use Machine Learning to Put a Botanist in Your Pocket.* TDS Archive. https://medium.com/data-science/plantsnap-puts-a-botanist-in-your-pocket-dc0d3f7aef47

Sohoni, N. S., Aberger, C., et al. (n.d.). *Low-Memory Neural Network Training: A Technical Report.* https://arxiv.org/pdf/1904.10631.pdf

World Leading Plant Identification Technology (n.d.). Plantsnap. https://www.plantsnap.com/