RU Recycle: A Smart Self-Sorting Trash Can Using Deep Neural Network

Steven Coulter, Umama Amhed, Yiwen Zhou, Jake Rodin, Professor Sheng Wei

May 8, 2019

Contents

1 Introduction

In the US we produce a significant amount of waste but only recycle a small amount of it. According the EPA, "In 2013, Americans generated about 254 million tons of trash and recycled and composted about 87 million tons of this material, equivalent to a 34.3 percent recycling rate. On average, we recycled and composted 1.51 pounds of our individual waste generation of 4.40 pounds per person per day," (EPA, 2016). That 254 million tons of trash can be sorted by type of waste, shown in this plot.

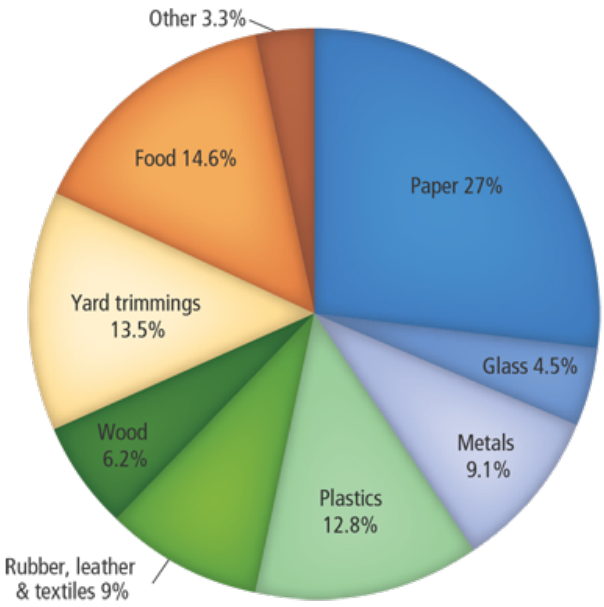

Figure 1: Percentage of Types of Waste Produced in 2013 (EPA,2016)

It clearly illustrates that the US disproportionally disposes of a lot of recyclable material as garbage. In total 53.4 percent of our waste in 2013 was recyclable but we only recycled 34.3 percent of it. This disparity is due to many factors, according to a 2016 article in the Huffington Post, The three major responses, as illustrated in the plot below, on why people do not recycle in order from most to least common are: people seeing recycling is inconvenient, it takes too much time and forgetfulness. These excuses have greatly hindered our efforts to effectively recycle.

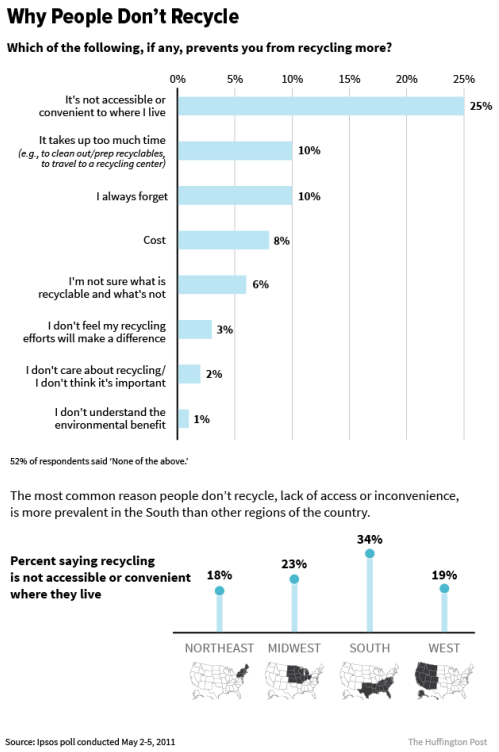

Figure 2: Responses to Why People Don't Recycle (Schumaker, 2016)

In response to these problems of inconvenience, we created a self-sorting trash can. This solution would make it easy for the user by decreasing time and eliminating the "thought" that goes to sorting waste, as well as cut down on the amount of menial labor required at recycling plants. This device will use deep learning and various sensors to categorize the disposed item and send it to the appropriate bin using a conveyor belt mechanism. The neural network we fine tuned uses thousands of images of waste to categorize its findings into paper, plastic, metal, glass, and trash. We will use a Raspberry Pi 3 B+ in conjunction with an Arduino Pro Micro to control a conveyor belt system. With our project, we aim to streamline the process of recycling while maximizing efficiency and accuracy to a degree that other similar products do not.

An example of a similar product would be CleanRobotics' TrashBot, which "utilizes a variety of sensors, robotics and artificial intelligence (AI) in order to detect what a piece of waste is, classify it as recyclable or not, then separate the item into the correct receptacle," (Greenwalt, 2017). Our project intends to take it a step further and sort the waste into five separate categories of paper, plastic, metal, glass, and trash. By having all the waste sorted before it reaches the recycling plant, our project will help decrease the amount of time, labor, and money expended by either users at home or recycling plants. This project will streamline the process of recycling while also decreasing the amount of time money and effort needed to recycle.

2 methods

As mentioned before, our goal was to create a device that uses deep learning and various sensors to categorize an item and send it to the appropriate bin through a conveyor belt mechanism. We used a variety of methods in order to successfully implement our design. We tackled this project from two engineering aspects: hardware and software.

2.1 Flowchart

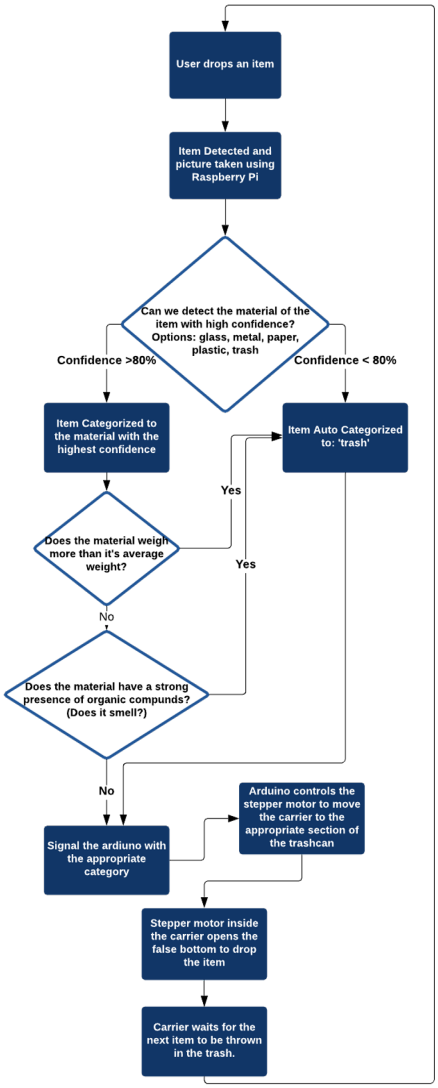

Figure 3: a flowchart detailing the project

2.2 Methods Used

2.2.1 Hardware

Before we started this project, we had to familiarize ourselves with some of the technologies we plan to use. From a hardware perspective, we needed to know how to use stepper motors, limit switches, and general pcb design. Some other skills that are required is a basic knowledge of soldering, constructing objects with power tools, and operating a 3D printer, which allows for faster acquisition of parts at a low cost. Experience with serial connections and C allowed for better use of the Arduino and bridging to the PI. Understanding and use of a laser cutter was also a necessary skill in order to produce the parts necessary for the projects.

2.2.2 Software

From the software side, the following were our design goals:

1. Use an appropriate machine learning architecture to finetune with our dataset.
2. Find a dataset to train given our recycling categories.
3. Pick a programming language that allows for easy interface with the Raspberry Pi and allows for easy use of neural network programming.

In addition, we had to be well acquainted with unix operating systems, crontab operations, and shell scripting in order to save power and run our system efficiently.

2.3 Design Implementation and chalanges

2.3.1 Hardware

2.3.1.1 Design

The device operates on the principle of linear control through a stepper - belt mechanism and control through an Arduino and Raspberry Pi mechanism. If the Raspberry Pi sends data to the Arduino, then the Arduino will respond by controlling the steppers to move over the top of the correct trashcan segment and have a servo drop the trash. The stepper will move back and forth to shake any excess trash off the platform. The

platform will then reset and the carrier will move back to the home position. This creates a feedback loop that is illustrated in the above flowchart.

2.3.1.2 Challenges

One of the largest challenges that we have encountered from the hardware perspective is the design constraint of the physical trash can. As it stands we have currently solved this by merging two existing trash cans; however ordering the chosen cans has proven difficult. Due to time constraints, we are forgoing creation of a specialized control unit that is capable of both control of the can and image analysis, such a control unit would be more effective in a large scale project however in the interest of the prototype, Thus in favor of both time and money, we have chosen to use a two board system, consisting of a Raspberry Pi and an Arduino. Overall we have not run into many regulatory issues, other than recycling standards and sorting codes. However, we expect the nuances of recycling code impede us in the future. Additionally, conventional trash cans available in stores are not large enough to house five types of trash in a realistic situation. Solution: merge two trash cans together to give more room to each compartment. The synchronization of the stepper motors caused a big challenge in the hardware side. In order to make them turn in sync, they had to be driven from the same Arduino pin. This caused issues, as one stepper driver originally did not use as much current as the other, causing issues in the feed torque and forcing the carrier to have a torque along the rods. Solution: tune the current output of the steppers using a multimeter.

2.3.2 Software

2.3.2.1 Design

From the software side, we wrote two python scripts to train and test our machine learning architecture and one final script to detect objects with machine learning and the sensors combine. Once an item is thrown into our device, we take a picture using Raspberry Pi camera and calculate the sorting category using the test code. If we can not determine a suitable category with at least 70% confidence, we will auto assign the item to the 'trash' category. Due to the fact that a wide variety of items can be thrown in trash, sometimes it can be challenging even for humans to decide if an item can be recycled or not. To avoid cross contamination, we do not want to throw things into one of the recycling categories unless we are absolutely confident of the resulting category. Furthermore, we had to find a database of images of trash that suited our needs. From our research online, we found the Trashnet, which was the most complete open source database that included variety of pictures from all our categories. We obtained permission from the creators before continuing on

with our project. We had to modified the database a little to include cardboard into paper category. "The pictures were taken by placing the object on a white poster board and using sunlight and/or room lighting. The pictures have been resized down to 512 x 384, which can be changed in data/constants.py (resizing them involves going through step 1 in usage). The devices used were Apple iPhone 7 Plus, Apple iPhone 5S, and Apple iPhone SE" (Thung, 2017).

2.3.2.2 Code Layout

In the following paragraphs, we will describe how we fine-tune our architecture to detect our recycling categories. We have chosen to work with ResNet50. While there are a lot of CNNs out there, a problem with directly mapped networks was that the deeper the networks were, the accuracy saturated and degraded more rapidly. ResNet uses network layers to fit a desired underlying mapping. It is easy for a residual block to learn an identity function, thus it allows us to add more and more layers to our training blocks without degrading quality of increasing training error (Jay, 2018). ResNet also uses a lot of the same convolutions so the dimensions between layers stay consistent. ResNet50 is a 50-layer Residual Network but there are other variants like ResNet101 and ResNet152 also. Below is an example of a residual block of ResNet: Below we

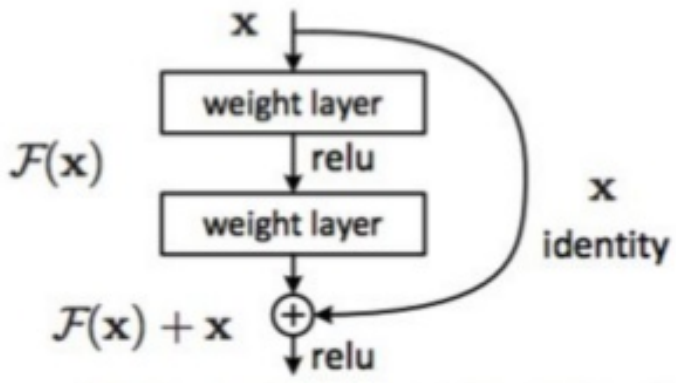

Figure 4: Residual block (Jay,2018)

will highlight the step-by-step process to fine tuning:

1. Preprocessing: keras includes a built-in image data generator which generates batches of image data with real time data augmentation looped over in batches. This is executed to suppress unwanted distortions or enhance some image features important for the next level. We use the flow_from_directory() function to specify our training directory. We set our class mode to categorical to categorize the 5 classes, set shuffle to true to randomize our images, and set our batch size to 16. Batch size defines the

number of samples propagated through the network: typically networks train faster in mini batches so we have chosen the value to be 16.

2. Fine-tuning Layers: After loading up ResNet 50, we have chosen to truncate the last layer with model.layers.pop(). Since we are only classifying 5 categories, we are replacing the layer with our own softmax layer so instead of the 1000 categories of imagenet, the new layer will only look at 5. From research we have found that it is common practice to freeze the previous layers of the pre-trained network. The initial layers extract universal features like curves and edges that are still relevant to the new classes, so we have chosen to not modify them. Another parameter to define here is the learning rate: it is simply the rate at which the network abandons old weights for new ones. we chose a very small learning rate of 0.0001, since the pre-trained weights are already pretty good, we don't want to distort them too much.

3. Model Fitting: we use the keras function fit_generator() to start the fine-tuning process. The number of training steps is specified to be training samples/batch size. We use a total of 2500 pictures to train our 5 classes and 30 images per category to validate. The term epoch refers to the number of times the entire dataset is passed forward and backward through the neural network. As the number of epoch increases, the number of times the weight changes increases as well. Now this could be a good thing and a bad thing: the good news is more epoch will result in underfitting whereas too many epochs can cause overfitting. The term overfitting means a network will be so good at recognizing your own dataset that it won't be able to recognize new images with slightly different parameters. Underfitting will give us a network that is too generalized to accurately recognize any images. To solve this issue we will use a method called early stopping.

```
early_stopping = EarlyStopping(monitor='val_loss',patience=10)
checkpointer = ModelCheckpoint('resnet50_best_train_epoch_50.h5', verbose=1, save_best_only=True)
```

Figure 5: Early Stopping function

Setting the patience to 10, we have basically setup a checkpoint to test if the loss has improved in the last 10 epoch. We set up save_best_only to true to save the best model after each checkpoint.

2.3.2.3 Challenges

Due to the fact that a wide variety of items can be thrown in trash, it can be challenging even for humans to decide if an item can be recycled or not. For example, containers with food/liquid still inside should not be recycled in order to avoid cross contamination. Solution: Add weight sensor and Volatile Organic

Compound (VOC) sensor to the carrier. These sensors will help determine if food particle/excess liquid is still inside a container. For example, if a plastic bottle is thrown into our device with liquid still inside, we would be able to use the weight sensor to determine that the object weighs more than an average plastic bottle, hence it belongs to trash. Additionally, if a greasy napkin is thrown into the device, we would be able to use the VOC sensor to detect the food particles and send the item to trash.

2.4 Tools used:

- Android Studio
- Oracle VM Virtual Machine on our windows 10 to use Ubuntu (64 bit)
- Python 2.7
- ResNet50
- trashnet
- Laser Cutter
- arduino IDE
- EasyTerm.

2.5 Limitations

2.5.1 Hardware

A Raspberry Pi is not quite fast enough for simultaneous use of the neural network and drive stepper motors. Secondarily, we have run into the issue of availability of some parts, namely the trashcans and the steel rods. However most of the technology we need is consumer grade technology and off the shelf CNC hardware, so it was easily implimented.

2.5.2 Software

We only have about 2500 images from trashnet, which is enough for now but will not be enough if this device was used at a large scale. So what we plan to do is continue to grow the database by uploading the picture

of any trash that we come across in our device, then once we have x amount of pictures, we will retrain our architecture.

2.6 Use of Standards

- IEEE 802.11 - Wifi, our project will employ wifi in order to communicate data back to our main servers.
- RS-232 - in order to enable communication between the Raspberry Pi and Arduino use of serial communication is required.
- RESNET50 - See Above paragraph on use of neural net
- Python 2.7 - Used to facilitate resnet50
- Tensorflow: Used to facilitate resnet50
- IEEE 1666-2005 - C standard library package, useful for arduino, 2005 version still relevant though newer versions available
- IEEE Standard 1003.1 - Bash standard, useful to limit power consumption on the pi because running the whole python script (having the entire neural network in ram at all times) would consume a lot of power.

2.7 Experiment/Product results

2.7.1 Hardware

The system functioned as intended. The hardware was able to move to the correct due to application of method of successive approximations. Below are some pictures of the functional hardware.

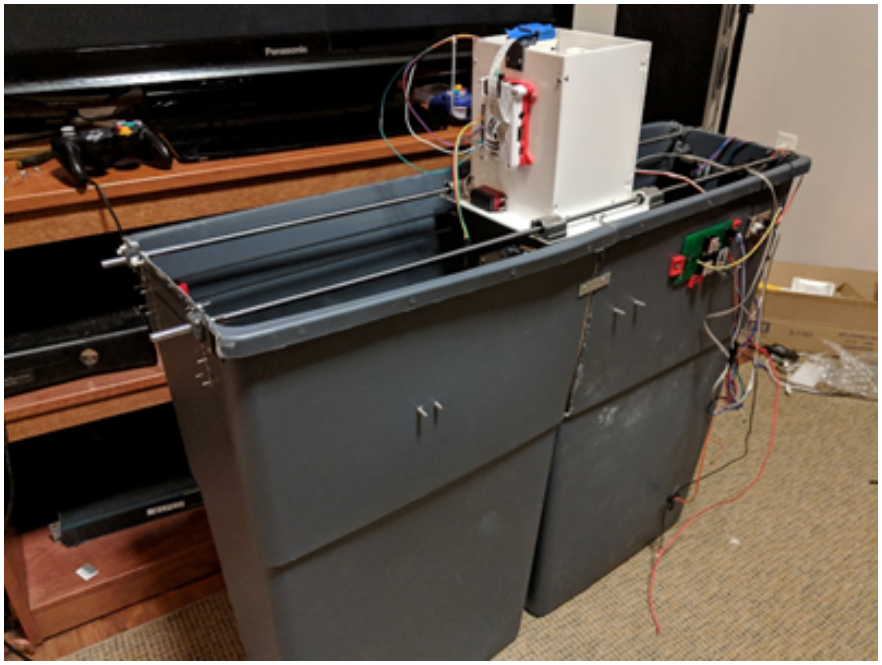

Figure 6: Trash can full view

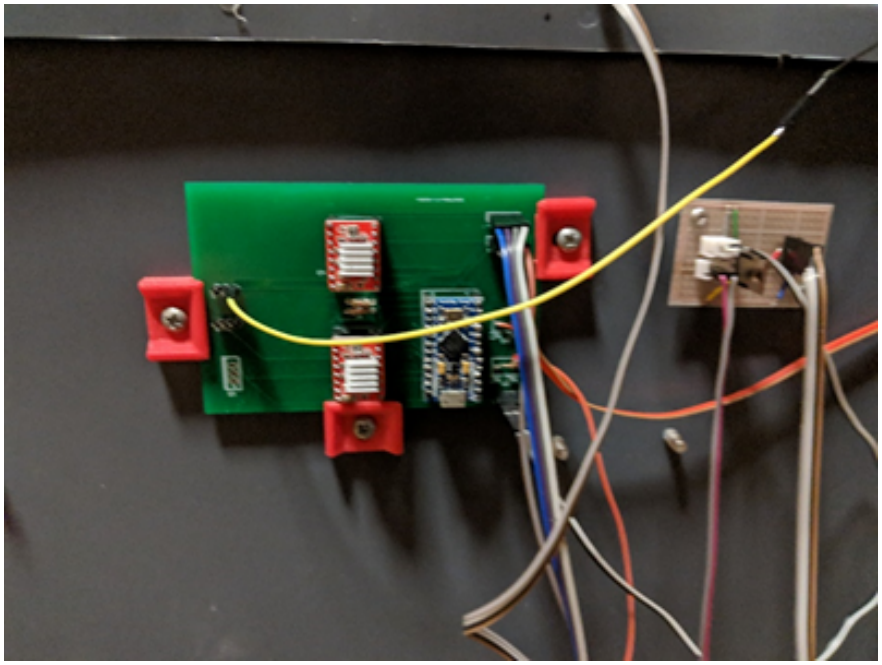

Figure 7: Conveyor belt control circuit

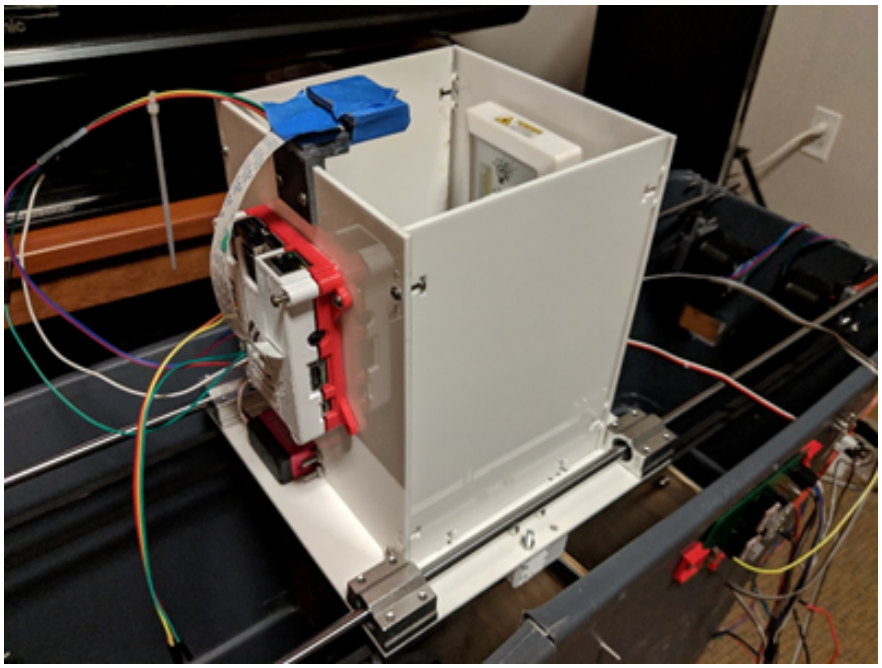

Figure 8: Acrylic carrier

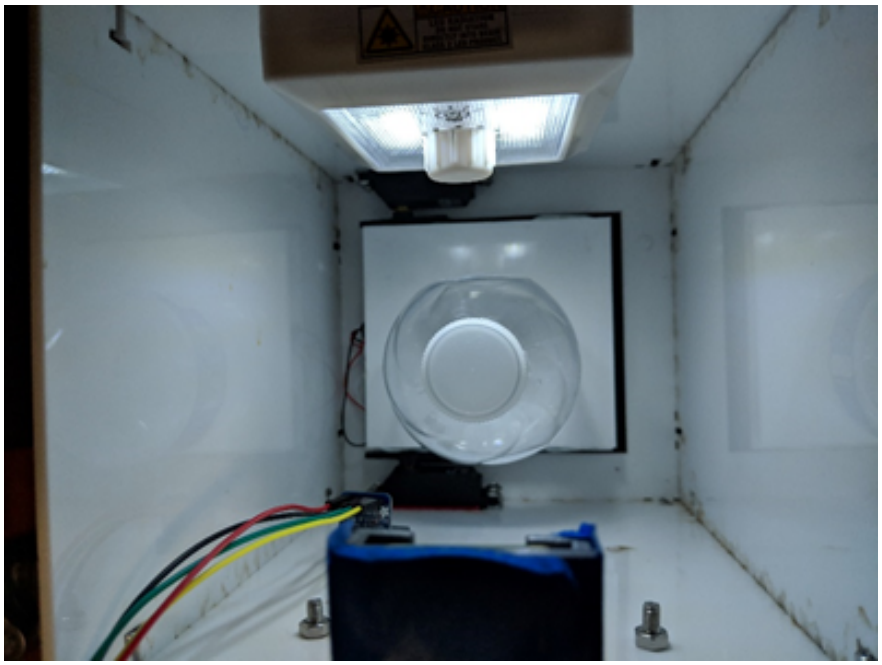

Figure 9: Carrier top view with trash inside

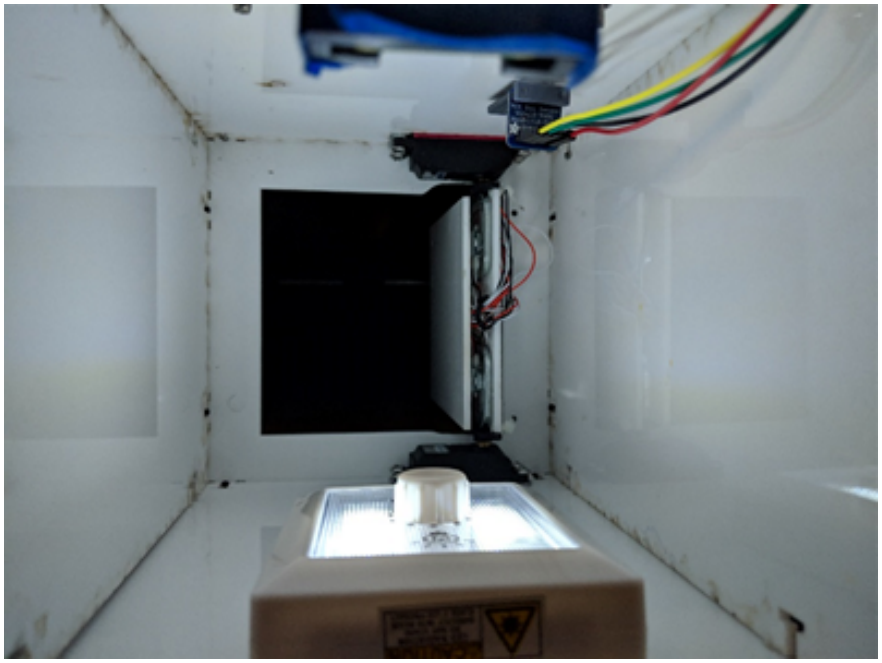

Figure 10: Carrier top view with false bottom open

2.7.2 Software

After training against the trashnet, I used 30 pictures of each category using the Raspberry Pi camera to test my prediction algorithm. I used sklearn to collect the accuracy, precision, recall, f-score to create my classification report. I also collected the accuracy of my prediction algorithm as seen below:

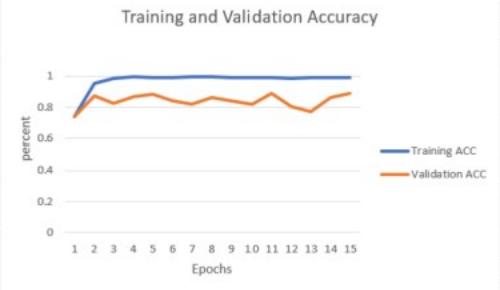

Figure 11: Training and Validation Accuracy vs the number of epochs

Materials	Precision	Recall	F-1 Score	Support
Glass	0.73	0.84	0.78	32
Metal	0.8	0.52	0.53	31
Paper	0.76	0.7	0.72	31
Plastic	0.62	0.52	0.56	31
Trash	0.42	0.58	0.5	31
Avg	0.7275	0.645	0.6475	31.25

Figure 12: Classification Report

As observed, glass, metal and paper was accurately validated over 70% of the times. Plastic was validated 62% of the time. Trash generated the worst result with 42% precision. The reason for precision of trash being significantly lower than the others is because the category itself is very diverse. Many things can be categorized as trash: some that may seem like they belong in one of the other categories at first glance. Therefore, we used the VOC sensor and weight sensor to decrease our errors. With more pictures of disposable items and a processing unit and camera more powerful than those provided by the Raspberry Pi, we would be able to increase our accuracy even more.

3 Cost and Sustainability Analysis

3.1 Economics (cost) impact

Our prototype costs $277.96. However, the price can be greatly reduced if we were to move our product into mass production. Buying items, such as metal rods, kevlar belts, and cables, in bulk would greatly reduce the cost for each individual bin. Currently, we are modifying premade trash cans to our specific design; although, if we were to employ a factory to make bins with our specific dimensions in large quantities, it would again greatly reduce the price and time to produce our product. The majority of costs for our electrical components can also be cut if we were to mass produce specific components for our trash can. This product is also energy efficient. When the trash can is not sorting, it turns off, so it does not draw power when it is not needed. According to a study from 2001 released by CalRecycle (formerly the California Integrated Waste Management Board), recycling one ton of waste would be equal to a pay increase of "$101 in salaries and wages, produce $275 more in goods and services, and generate $135 more in sales than disposing of it in

a landfill" (Morrigan, 32). Using these numbers, if half of the recyclables and compostables from the west coast (California, Oregon, and Washington) reported in this study were recycled, then the result would be "almost a $1.6 billion increase in additional salaries and wages, $818 million increase in additional goods and services produced, and $309 million increase in additional sales" (Morrigan, 32). These numbers would also translate to increased revenue for each state, especially through income, property, and sales taxes (Morrigan, 32). Of course, if we calculated this on a national number, the numbers would be even bigger. In fact, the EPA (Environmental Protection Agency) has done these calculations in an update to the national Recycling Economic Information (REI) Study in 2016. The study found that a single year of recycling in the United States accounted for 757,000 jobs, $36.6 billion in wages, and $6.7 billion in tax revenues ("Recycling Basics", 2018).

3.2 Environmental impact of the product

Our autonomous sorting trash should provide a positive impact to the environment. The average US citizen discards 7.5 pounds of solid waste a day ("Frequently Asked Questions: Benefits of Recycling", 2019). By removing recyclables from the solid waste stream, we save a lot of space in landfills, preventing them from overflowing too quickly. Recycling also slows down our consumption of natural resources like wood and water, prevent pollution because we are reducing the need to collect raw materials, and save the energy needed to manufacture new products ("Benefits of Recycling," 2019). For example, recycled aluminum saves 95% production energy, recycled steel 60%, recycled newspapers 40%, recycled plastics 70%, and recycled glass 40% ("Benefits of Recycling," 2019). In 2010, to meet the growing paper consumption demands, 80% more wood would need to be harvested in comparison to only 20% more wood needed if people were actively recycling paper ("Benefits of Recycling," 2019). Recycling reduces greenhouse gas emissions too ("National Overview: Facts and Figures on Materials, Wastes and Recycling", 2018).

3.3 Social impact of the product

Our product will make recycling less labor intensive and more convenient for the average consumer. It will cut down the time and energy needed for everybody to recycle and the need for manual recycling in single stream recycling facilities. It also reduces the need for landfills, which would make towns prettier and less smelly on days where the wind blows in the wrong direction.

4 Conclusion

Our goal for this project was to make the process of recycling more accurate and efficient at the bin level. We were able to do so by creating a device that sorts and disposes of items thrown with high accuracy. We fine tuned the Resnet-50 neural network architecture with thousands of images of waste to categorize its findings into: paper, plastic, metal, glass, and trash. We used a Raspberry Pi 3 B+ in conjunction with an Arduino Pro Micro to control the conveyor belt system. We were able to achieve a precision score of 72% on average among the 5 categories. Trash being a complex category, achieved the lowest precision score, so we improved our results by adding a VOC sensor and a weight sensor to detect leftover food in the disposed item. In the future, we would like to use a more powerful processing unit than Raspberry pi to decrease operation time at the software level. We want to collect more pictures of disposable items to improve our fine tuning architecture. From the hardware side, we would like to build a bigger acrylic carrier to allow for larger items to be disposed. Additionally, we plan to experimentally determine the location of the bins and rotation of the belts to eliminate any possibility of dropping an item in the wrong section of the trash can.

5 Apendix

Attached below is our code for the project:

5.1 predict.py

the live runnable for our code demo On the raspberry Pi

```
import json, os, re, sys
import numpy as np
from keras import backend as K
from keras.models import load_model
from keras.preprocessing import image
from picamera import PiCamera
from time import sleep
from PIL import Image
import serial
import board
import busio
import adafruit_sgp30
import time

#Initiate weight sensor
EMULATE_HX711=False
if not EMULATE_HX711:
```

```
    import RPi.GPIO as GPIO
    from hx711 import HX711
else:
    from emulated_hx711 import HX711

def cleanAndExit():
    print ("Cleaning...")

    if not EMULATE_HX711:
        GPIO.cleanup()

    print ("Bye!")
    sys.exit()

hx = HX711(5, 6)

if __name__ == '__main__':
    model_path = sys.argv[1]

    print ('Loading serial')
    for x in range(10):
        name = '/dev/ttyACM' + str(x)
        try:
            ser = serial.Serial(name)
        except:
            continue

    print(ser.name)

    print('Loading model:', model_path)
    model = load_model(model_path)
    image2 = 'image2.jpg'
    run= 'y'


    voc = True
    weight_sensor = True

    # Setup i2c
    print('Setting up i2c...')
    try:
        thres_smell = 0
        min_smell = 0
        max_smell = 0
        smell = []
        i2c = busio.I2C(board.SCL, board.SDA, frequency=100000)
        sgp30 = adafruit_sgp30.Adafruit_SGP30(i2c)
        sgp30.iaq_init()
        sgp30.set_iaq_baseline(0x8973, 0x8aae)
        for i in range(40):
            print("eCO2 = %d ppm \t TVOC = %d ppb" % (sgp30.eCO2, sgp30.TVOC))
            sleep(1)
    except:
        print ("VOC failed")
```

```
    voc = False

# Setup weight sensor
print('Setting up weight sensor...')
try:
    hx.set_reading_format("MSB", "MSB")
    hx.set_reference_unit(4.09)
    hx.reset()
    hx.tare(50)
    offset_prev = hx.tare(50)
except:
    print ("weight sensor failed")
    weight_sensor = False
print ("Place object now...")

while(run =='y'):
    thres_smell = 0
    camera = PiCamera()
    camera.start_preview()
    sleep(10)
    camera.capture(image2)
    camera.stop_preview()

    if voc:
        smell = []
        for i in range(20):
            eCO2 = sgp30.eCO2
            TVOC = sgp30.TVOC
            print("eCO2 = %d ppm \t TVOC = %d ppb" % (eCO2, TVOC))
            smell.append(TVOC)
            sleep(0.25)
        smell = np.array(smell)
        b = smell[::-1]
        max_smell = len(b) - np.argmax(b) - 1
        if (max_smell == 0 ):
            thres_smell = 0
            min_smell = max_smell
        else:
            min_smell = np.argmin(smell[0:max_smell])
            thres_smell = smell[max_smell] - smell[min_smell]
        print("max", smell[max_smell], "min", smell[min_smell])

    imageCrop = Image.open(image2)
    cropped = imageCrop.crop((570,105,1365,830))
    save = cropped.save('imageC.jpg')
    img2 = image.load_img('imageC.jpg', target_size=(224, 224))
    y = image.img_to_array(img2)

    hx.set_offset_A(offset_prev)
    y = np.expand_dims(y, axis=0)
    preds = model.predict(y)
    print(preds)
    print('Generating predictions on image:', image2)
```

```
        category = 4
        for p in range(4):
            if (preds[0][p] >= 0.60):
                category = p


        target_names = [b'glass\n',b'metal\n',b'paper\n',b'plastic\n',b'trash\n']
        print('Predicted Category: ')
        print(target_names[category])
        print('After VOC test')
        if (thres_smell >= 20) and voc:
                category = 4
                print('VOC smelled food')
        print(target_names[category])

        weight = np.zeros(10)
        for w in range(10):
            weight[w] = hx.get_weight(1)
        med_weight = np.median(weight)
        print("Weight (in grams) =", med_weight)
        if (category == 1 or category == 3):
            print('After food weight test')
            if (med_weight >= 200) and weight_sensor:
                    category = 4
        print(target_names[category])
        ser.write(target_names[category])


        run = input('would you like to continue? y/n')
        camera.close()
        offset_prev = hx.tare(50)
        sleep(5)
    ser.close()
```

5.2 train.py

Training code for the computer vision system.

```
import math, json, os, sys

import keras
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.layers import Dense, AveragePooling2D
from keras.layers import Flatten
from keras.models import Model
from keras.optimizers import Adam
from keras.preprocessing import image
import numpy as np
import sys
import scipy
from scipy import signal
from keras.applications.resnet50 import ResNet50
```

```
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
import warnings



DATA_DIR = 'data'
TRAIN_DIR = os.path.join(DATA_DIR, 'train')
VALID_DIR = os.path.join(DATA_DIR, 'valid')
SIZE = (224, 224)
BATCH_SIZE = 16

if __name__ == "__main__":

    num_train_samples = sum([len(files) for r, d, files in os.walk(TRAIN_DIR)])
    num_valid_samples = sum([len(files) for r, d, files in os.walk(VALID_DIR)])

    num_train_steps = math.floor(num_train_samples/BATCH_SIZE)
    num_valid_steps = math.floor(num_valid_samples/BATCH_SIZE)

    gen = keras.preprocessing.image.ImageDataGenerator(rotation_range = 45,
        horizontal_flip=True, vertical_flip=True)
    val_gen = keras.preprocessing.image.ImageDataGenerator(horizontal_flip=True,
        vertical_flip=True)

    batches = gen.flow_from_directory(TRAIN_DIR, target_size=SIZE, class_mode='
        categorical', shuffle=True, batch_size=BATCH_SIZE)
    val_batches = val_gen.flow_from_directory(VALID_DIR, target_size=SIZE, class_mode=
        'categorical', shuffle=True, batch_size=BATCH_SIZE)
    input_size = 224
    input_channels= 3



    classes = list(iter(batches.class_indices))
    base_model = ResNet50(include_top = False,weights="imagenet",input_shape=(
        input_size,input_size,input_channels))
    x = AveragePooling2D((7,7),name='avg_pool')(base_model.output)
    x = Flatten()(x)
    x = Dense(len(classes),activation="softmax")(x)
    finetuned_model = Model(base_model.input,x)
    for layer in base_model.layers:
        layer.trainable = False
    finetuned_model.save_weights("all_nontrainable50.h5")

    base_model = ResNet50(include_top = False,weights="imagenet",input_shape=(
        input_size,input_size,input_channels))
    x = AveragePooling2D((7,7),name='avg_pool')(base_model.output)
```

```
    x = Flatten()(x)
    x = Dense(len(classes),activation="softmax")(x)


    finetuned_model = Model(base_model.input,x)

    for layer in base_model.layers[:10]:
        layer.trainable = False
    finetuned_model.load_weights("all_nontrainable50.h5")

    finetuned_model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy',
         metrics=['accuracy'])
    for c in batches.class_indices:
        classes[batches.class_indices[c]] = c
    finetuned_model.classes = classes

    early_stopping = EarlyStopping(monitor='val_loss',patience=10)
    checkpointer = ModelCheckpoint('resnet50_best_train_epoch_50.h5', verbose=1,
        save_best_only=True)

    history = finetuned_model.fit_generator(batches, steps_per_epoch=num_train_steps,
        epochs=50, callbacks=[early_stopping, checkpointer], validation_data=
        val_batches, validation_steps=num_valid_steps)
    finetuned_model.save('resnet50_train_epoch_50.h5')
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    print(acc)
    print(loss)
    print(val_acc)
    print(val_loss)
    ep = range(len(acc))
    plt.plot(ep, val_loss,'b',label = 'Valid loss')
    plt.plot(ep,loss,'r',label='Training loss')
    plt.plot('T and V Loss')
    plt.legend()
    plt.show()
```

Code running on the arduino to preform the motor control operations.

5.3 arduinocode.ino

```
#define dir 7
#define sw1 8 // MUST BE PULLED HIGH FOR SWITCHES to work
#define sw2 9
#define en A1
#define stepp A0
#define  servo A2
#include <Servo.h>
Servo myservo;
const int stepsPermm = 10;  // change this to fit the number of steps per revolution
```

```
int currentpos = 0;
String incoming = "";
// for your motor

// initialize the stepper library on pins 8 through 11:
enum input {glass, paper, plastic, metal, trash, bad};
input readin (String const& inString) {
  if (inString == "glass\n") return glass;
  if (inString == "paper\n") return paper;
  if (inString == "plastic\n") return plastic;
  if (inString == "metal\n") return metal;
  if (inString == "trash\n") return trash;
  return bad;
}

void moveto (int pos) {
  digitalWrite(en, LOW);
  digitalWrite(dir, LOW);
  digitalWrite(stepp, LOW);
  int stepon = 0;
  while (digitalRead(sw2) == HIGH) {
    if (stepon == 0) {
      stepon = 1;
      digitalWrite(stepp, LOW);
      delay(2);
    } else {
      stepon = 0;
      digitalWrite(stepp, HIGH);
      delay(2);
    }

  }
  delay(100);
  digitalWrite(dir, HIGH);
  for (int x = 0; x < pos * stepsPermm; x++) {
    if (stepon == 0) {
      stepon = 1;
      digitalWrite(stepp, LOW);
      delay(2);
    } else {
      stepon = 0;
      digitalWrite(stepp, HIGH);
      delay(2);
    }
  }
  for(int x = 10; x <= 120; x++){
        myservo.write(x);
    delay(15);
  }
  delay(3000);
  for(int x = 120; x >= 10; x--){
    myservo.write(x);
    delay(15);
  }
```

```
  delay(1500);
  digitalWrite(dir, LOW);
    while ( digitalRead(sw2) == HIGH) {
    if (stepon == 0) {
      stepon = 1;
      digitalWrite(stepp, LOW);
      delay(2);
    } else {
      stepon = 0;
      digitalWrite(stepp, HIGH);
      delay(2);
    }

  }
  digitalWrite(en, HIGH);
Serial.println("Ok");
}
void setup() {

  pinMode(sw1, OUTPUT);
  pinMode(sw2, INPUT);
  pinMode(dir, OUTPUT);
  pinMode(en, OUTPUT);
  pinMode(stepp, OUTPUT);
  myservo.attach(servo);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(sw1, HIGH);
  digitalWrite(en, HIGH);
  incoming = "";
  // step one revolution  in one direction:
  if (Serial.available() > 0) {
    // read the incoming byte:
    incoming = Serial.readString();
    switch (readin(incoming)) {
      case metal:
      Serial.println("movining to metal");
        moveto(0); // for saftey
        break;
      case paper:
        Serial.println("movining to paper");
        moveto(130);
        break;
      case glass:
      Serial.println("movining to glass");
        moveto(250);
        break;
      case plastic:
      Serial.println("movining to plastic");
        moveto(415);
        break;
      case trash:
```

```
      Serial.println("movining to trash");
        moveto(600);
        break;
      default:
        break;
    }
  }
}
```

6 Works Cited

(n.d.). Retrieved from https://www.gnu.org/software/bash/manual/html$_node/What-is-Bash_03f.html$

1666-2005 - IEEE Standard SystemC(R) Language Reference Manual. (2006, March 31). Retrieved from https://ieeexplore.ieee.org/document/1617814

Benefits of Recycling. (n.d.). Retrieved from https://nems.nih.gov/environmental-programs/Pages/Benefits-of-Recycling.aspx

Frequently Asked Questions: Benefits of Recycling. (n.d.). Retrieved from https://lbre.stanford.edu/pssistanford-recycling/frequently-asked-questions/frequently-asked-questions-benefits-recycling Greenwalt, M. (2017, July 27). One Pittsburgh-based Tech Company Has Developed a Self-Sorting Trash Bin. Retrieved from https://www.waste360.com/fleets-technology/one-pittsburgh-based-tech-company-has-developed-self-sorting-trash-bin

Jay, P., Jay, P. (2018, February 07). Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image... Retrieved from https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624

Morrigan, M., Smith, B., Davis, J. (2011). Reducing Greenhouse Gas Emissions through Recycling and Composting in California, Oregon, and Washington. The Evans School Review, 1(1), 23-35. Retrieved February 10, 2019, from https://depts.washington.edu/esreview/wordpress/wp-content/uploads/2012/12/ESR-2011-Research-Reducing-Greenhouse-Gas-Emissions-Through-Recycling-and-Composting.pdf

Municipal Solid Waste. (2016, March 29). Retrieved from https://archive.epa.gov/epawaste/nonhaz/municipal/web/html/

National Overview: Facts and Figures on Materials, Wastes and Recycling. (2018, October 26). Retrieved from https://www.epa.gov/facts-and-figures-about-materials-waste-and-recycling/national-overview-facts-and-figures-materialsGenerationTrends

Recycling Basics. (2018, August 01). Retrieved from https://www.epa.gov/recycle/recycling-basics