1 Introduction

In this paper we will discuss an approach to detecting US household recyling bins and attempt to understand the shortcomings of the given implementation. Possible improvements will be suggested along the way alongside the state of the current results.

1.1 Importance of Bin Segmentation

The problem of detecting the relative location of a recycling bin has many novel uses in the modern span of problems. More precisely, a solution such as this one is immediately useful as part of the self-driving car problem to detect and avoid obstacles on the road such as these bins. Another possible utility could be to enable autonomous pickup and tracking of these bins for the cars that make their trips to empty them, possibly yielding an expedited route time.

1.2 Introduction to Approach

The problem at hand has been solved with a variety of tools to label, reject, and classify points within the images but fundamentally the core of the solution lies in the logistic regression used for classification of pixel class. With the classified image from the regression, all that is left to solve is bounding a box to the mask given some geometric and spatial filters such as aspect ratio.

2 Problem Formulation

2.1 Pixel Classification

2.1.1 Problem Statement

In this problem we are given labeled data with classes as $y \in \{red, green, blue\}$ or $y \in \{1, 2, 3\}$ respectively. We are tasked with classifying the test pixels into one of the three classes.

2.1.2 Metrics

There is one metric of performance that will be used which is accuracy defined as:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I(\hat{y}_i, y_i)$$

Where I is the indicator function, the functions is 1 if both arguments are the same and 0 otherwise. The hat on y implies the predicted classes and y is the ground truth classes. Note, this accuracy is only really important when evaluated on the test set.

2.2 Bin Detection

2.2.1 Problem Statement

In this problem we are tasked with establishing bounding boxes for recycle bins in images. More is detailed on the intermediate details and metrics in the technical approach but as an input:output scheme we are concerned with just the bounding boxes from the input image.

2.2.2 Metrics

Other than metrics used intermediately, such as pixel classification accuracy, the primary metric of performance is similarity of the ground-truth bounding boxes and the predicted bounding boxes. This is calculated as the intersection over union specified as:

$$IoU = \frac{\hat{B} \cap B}{\hat{B} \cup B}$$

This is already implemented for us but $IoU \geq 0.5$ is considered a success.

3 Technical Approach

3.1 Pixel Classification

I chose to use logistic regression to first try and classify one of three classes of pixels in the given data set of {red, green, & blue} pixels. I mostly chose this method due to my familiarity and enjoyment in working with it. Let's dive a bit deeper into how I formulated my model.

I will talk mostly in regards to the fundamental math behind logistic regression as compared to linear regression and how the optimization is formulated.

3.1.1 Linear vs Logistic Regression

The simplest way to describe the difference between two fundamentally similar models is when considering

usage. Logistic regression has all of the forward propagation as linear regression but induces a nonlinearity using the sigmoid as shown in Figure 1.

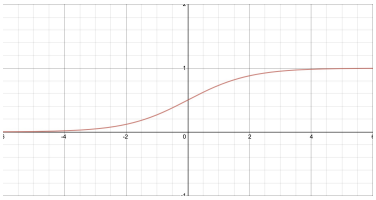

Figure 1: Sigmoid Function

As we can see the function is bounded as $y \in [0, 1]$ which makes the output suitable for probabilities. This lended itself well to classification with a decision boundary at $y = 0.5$. The inverse is also true, linear regression is much better suited to prediction of real-valued numbers.
Going back to our problem, logistic regression works well as we will attempt to classify each pixel of the image as belonging to a recycle bin or not. In the first case of the basic classification we actually have a multinomial logistic regression with a distinct logistic regression for each class and the prediction is chosen as the class that yields the highest probability.

3.1.2 Logistic Regression Predictions

So then, mathematically what does logistic regression look like? Let us call a vector $\mathbf{X}$ to represent a matrix of all of our sample inputs (1 sample = 1 pixel). If we take the i-th row of $\mathbf{X}$ and dot it with vector W of the same size (same number of features), we have the start of linear and logistic regression. We also need to shift the result by a bias that we will call b. This yields for linear regression the following:

$$\hat{y_i} = W^T \mathbf{X}_i + b$$

, and for logistic regression we apply the sigmoid function σ to yield:

$$\hat{y_i} = \sigma(W^T \mathbf{X}_i + b)$$

where $\hat{y}$ is the predicted value.
To further clarify, W is called the weight vector and it can be thought of as mapping the relation or sensitivity of the output to changes in particular features of $\mathbf{X}$. How does one determine the correct value of W? That is done with a process called training using the unconstrained optimization technique gradient descent.

3.1.3 Gradient Descent Training Process

The process of training is fairly simple, we cannot explicitly compute where the below derivative is all equal to 0:

$$\frac{\delta \mathcal{L}}{\delta W}$$

Where $\mathcal{L}$ is the loss function. The loss function gives us a way of computing the performance of the weights and bias on the training data set. For logistic regression a common loss function is:

$$\mathcal{L}(W) = -\frac{1}{N}\sum_{i=1}^{N} \log(p_i) = -\frac{1}{N}\sum_{i=1}^{N} \log \sigma(y_i W X_i)$$

Important note: I find it simpler to append a 1 to each row of $\mathbf{X}$ and increase the size of W by 1 to handle the bias in the dot product as opposed to a separate term.
Gradient descent follows a simple update rule that uses the derivative and iterates to attempt to converge to a minimum of the loss function. Knowing the derivative of the aforementioned loss function we get the update rule:

$$W_{i+1} \leftarrow W_i - \alpha \frac{-1}{N}\sum_{i=1}^{N} \log \sigma(-y_i W X_i) y_i X_i$$

Where α is a hyper-parameter called the learning rate that is tuned to increase the speed of learning but also meter unstable convergence.

3.1.4 Feature Engineering

As we would expect glare and other factors to affect the color of the pixels in the recycle bin images I preemptively added the HSV color space to the features for all pixels. HSV helps because even for a washed out region the value of the pixel can represent a blue pixel that is near 0 saturation. Looking at a particular sample (let's say the i-th sample) the features would be:

R	G	B	H	S	V	1

Where the 1 is so that the last weight in W can be the bias.

3.1.5 Performance on TriColor Set

Using the given labeled data set with three classes I trained 3 individual logistic regressions, one for each class, and presented below is the training curve for the blue pixels.

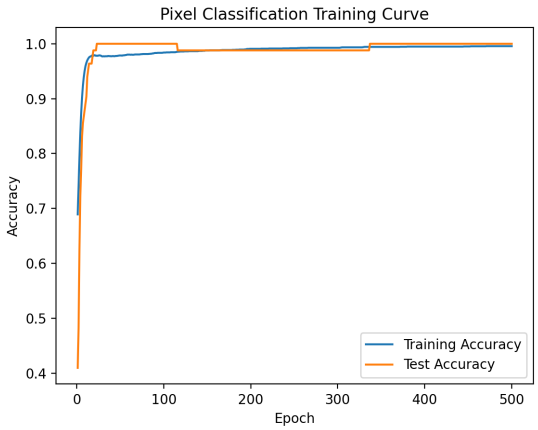

Figure 2: Training curve for the tricolor set

As we can see the accuracy is great, given this we will use this framework moving forward for the bin pixel classifications. We now just need the examples for $\mathbf{X}$ and y to train and test the model.

3.2 Recycle Bin Data Acquisition

For the prior dataset, we were given the examples and their labels, but for the bin classification we only have images that may or may not contain the bins that we are trying to detect.
Using the given code for roipoly, I added some small tweaks that allowed me to quickly build a labeled set of pixels from the images. I used the mask from roipoly to separate pixels for which $y_i = 1$ and those that are not recycle bins. To ensure that the data was well separated, I scattered data for $y_i = 1$ and $y_i = 1$ and made a collage of the pixels. We get the following:

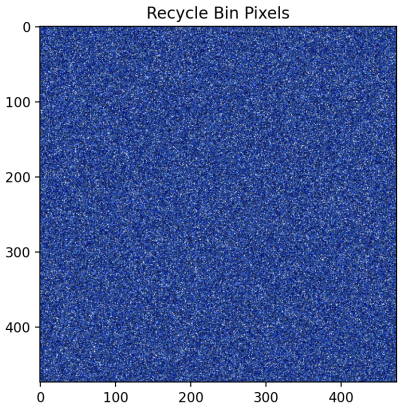

Figure 3: Collage of recycle bin pixels.

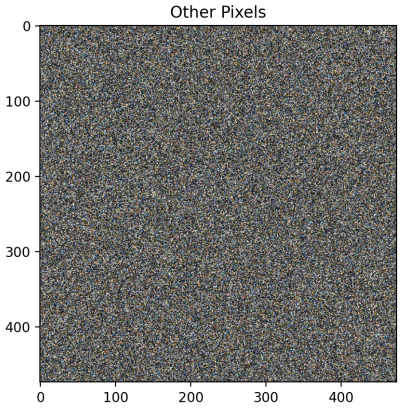

Figure 4: Collage of non recycle bin pixels.

As we can see there is a clear separation between the two sets, and we are in good shape to proceed with training with our prior framework.

3.3 Bin Color Training

There is not much to say here except that the results from training against the image shown in the collages is good. One change that was made from the prior framework was to use stochastic gradient descent (SGD). With such a large dataset, computing the gradient over all samples can be expensive. With SGD you randomly sample a smaller set on each iteration so that you progress through training faster and over many iterations sample the entire set. This is why we see a dramatic oscillation on the training set accuracy as the training set computed is merely the particular subset from the SGD on each iteration.

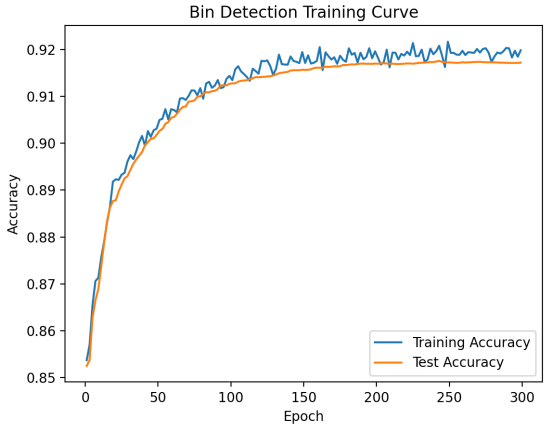

Figure 5: SGD training of bin pixel classification.

3.4 Bin Color Classification

Using the prior data set, I trained a single logistic regression to classify the pixels in the image. Recall that the predicted value from a logistic regression is a probability so we introduce a hyper-parameter to tune: image-threshold. This threshold determines the decision boundary for what is a recycle bin or not, we will tweak this later. Below if an example image with it's probabilities and example bounding boxes.

3.5 Bounding Box Placement

In terms of the bounding boxes it was fairly simple to implement. With the binary mask image like that of Figure 7, I merely needed to apply some filtering to improve the rejection of false positives and separate local matches. There were three main filter utilized that were tuned:

- Eroding: Before fitting a bounding box I used an erosion to attempt to separate lightly grouped boxes such as those in the figures ahead.
- Minimum Area: I imposed a minimum area required as to reject small groups across the mask.
- Aspect Ratio: After fitting the boxes, I check to see if they are within a range of aspect ratios.

Figure 6: Original image of bins.

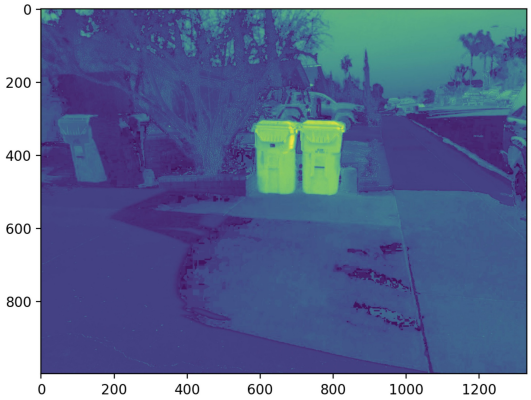

Figure 7: Probability image of prior image.

Figure 8: Bounding box on masked binary image.

4 Results

The results section serves to present the performance of the solution, but more importantly to identify failure modes and issues that may need further work.

4.1 Pixel Classification

4.1.1 Summary of Results

Across the 3 classes of validation data I locally scored 100% accuracy. When uploaded to gradescope for the test set we see a slightly lower overall accuracy of 98.8% which is still very good.

4.1.2 Final Parameters

Below are the 7 weights for the 3 class-specific logistic regressions. Note again, W_7 is the bias.

W_1	W_2	W_3	W_4	W_5	W_5	W_7
9.08	-5.02	-4.78	-0.09	-2.46	-1.35	-1.08
-5.48	8.65	-4.4	-1.14	-2.47	-1.43	-1.24
-5.5	-4.32	8.52	1.34	-2.43	-1.4	-1.31

4.2 Bin Detection

4.2.1 Summary of Results

After the gradescope submission, we see a score on the validation set of 7.75 which indicates that we have some mistakes. I will try to detail what I believe are some possibilities as to why there were false positives or false negatives.

4.2.2 Final Parameters

Below are the 7 weights for the single class logistic regression. Note again, W_7 is the bias.

W_1	W_2	W_3	W_4	W_5	W_5	W_7
-0.74	-0.32	0.92	0.97	1.54	0.58	-0.17

LR	Epochs	Weight Decay	Threshold	Disk Size
0.15	150	0.001	0.63	4

4.2.3 Example Segmentations

We cover plenty of issues regarding segmentation and have prior shown successful segmentations, so I will not repeat additional examples here.

4.2.4 Threshold Tuning

One of the important hyper-parameters is the classification threshold. Increasing this reduces the number of pixels classified as recycle bins which can help reject false positives but too high and you can lose the bin in the mask.

Figure 9: Example image for thresholding.

Figure 10: Example image for threshold = 0.63

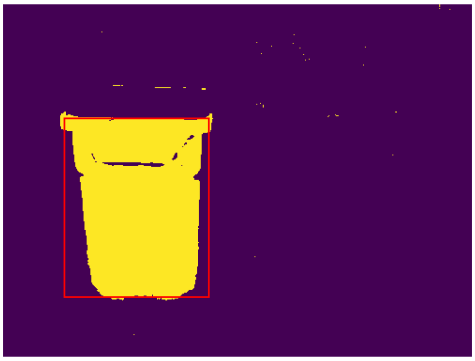

Figure 11: Example image for threshold = 0.55

As we can see with a more aggressive threshold we lose out on some of the detail and the bounding box is forced to be inset from the edge of the bin. For any particular image we can tune the threshold but we have to remember that we are looking to maximize the accuracy across all images and this image was an outlier.

4.2.5 Erosion Tuning

Another important hyper-parameter is the radius of the erosion applied to the image before finding the bounding boxes. Images with bins that are close to each other will often have small interconnects that force one larger bounding box or worse rejection of both. It's important to erode away the connection but often with too much erosion you can have similar issues to that of threshold tuning where the bounding box is inset.

Figure 12: Example image for erosion tuning.

Figure 13: Example image for erosion disk = 4

Figure 14: Example image for erosion disk = 2

In the above examples, we can see that the difference between an erosion disk of 2 and 4 completely changes the state of the bounding box, but with too much erosion it can destroy the quality of the individual boxes.

4.2.6 False Positives

Across the set it's likely to have false positives with simpler filters. Given my filters, there are cases where the disk radius could cause false positives. Below is one example:

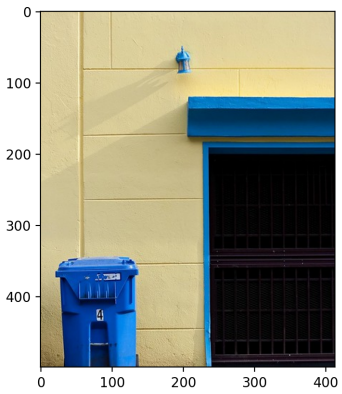

Figure 15: Example image for erosion tuning.

Figure 16: Example image for erosion disk = 4

Figure 17: Example image for erosion disk = 2

With a disk of radius 2 (which helped for other cases) the doorway is a false positive. One way to improve the accuracy is to improve the filters as shown here but that was not implemented here.

4.2.7 Bounding Box Coordinates

Img	MinR	MinC	MaxR	MaxC
0061	180	107	314	300
0062	228	184	413	499
0062	22	345	137	499
0063	173	96	266	226
0064	346	108	464	272
0065	751	415	932	621
0066	NA	NA	NA	NA
0067	587	306	832	510
0068	NA	NA	NA	NA
0069	NA	NA	NA	NA
0070	0	78	59	165
0070	0	156	231	364
0070	108	563	165	609

5 Acknowledgements

5.1 Karndeep Singh

Karndeep and I conferred on topics regarding bounding box filtering.

5.2 Piazza

All other references were from Piazza.