Project 1: Color Classification and Recycling Bin Detection

Collaboration in the sense of discussion is allowed, however, the assignment is individual and the work you do should be entirely your own. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276a. *Books, websites, and papers may be consulted but not copied from. It is absolutely forbidden to copy or even look at anyone else's code.* ***Please acknowledge in writing people you discuss the problems with.***

Submission

Please submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. **Programming assignment**: upload all code you have written for the project (do not include the training and validation datasets) and a README file with a clear, concise description of the main file and how to run your code. The Gradescope autograder will test your "pixel_classifier.py" and "bin_detector.py" on the test sets. You are allowed infinite number of attempts before the deadline.

2. **Report**: upload your report in pdf format. You are encouraged but not required to use an IEEE conference template[1] for your report. Please refer to Lecture 1 for the expected report structure and contents[2].

Problems

In square brackets are the points assigned to each part.

1. [25 pts] Train a probabilistic color model from pixel data to distinguish among red, green, and blue pixels. The data and starter code for this part is contained in the folder pixel_classification. The data consists of a training set and validation set. Each example in the training or validation sets is a 28×28 image with a single RGB value at all of its pixels. The images are split according to the three labels: red, green, blue. You must implement Logistic Regression or Naïve Bayes or Gaussian Discriminant Analysis for pixel color classification. You can implement either a trenary model (classifying one out of three classes) or three one-vs-all binary models. The provided files show how to collect the labeled training data in the standard format $X \in \mathbb{R}^{n \times 3}$ and $\mathbf{y} \in \{1, 2, 3\}^n$, where $1 =$ red, $2 =$ green, $3 =$ blue and n are the number of examples. Another file shows how to evaluate your model on the validation set. The algorithms you implement in this part will be helpful for and can be used directly in the next part of the project (of course, re-training with new/additional data will improve the blue classifier performance).

   You should use the provided starter code "pixel_classifier.py" and implement the function "classify()". For this file, please do not change the file name, class name, function names, or function arguments because they will be used by the Gradescope autograder. Also, **do not use any built-in functions that implement a core part of this project** (Logistic Regression, Naïve Bayes, Gaussian Mixtures, EM). If you are not sure, then ask the TAs if a particular package may be used.

2. [30 pts] Train a probabilistic color model to recognize recycling-bin blue color and use it to segment unseen images into blue regions. Given the blue regions, detect blue recycling bins and draw a bounding box around each one. See Fig. 1 for an example image, containing one blue recycling bin. The data and starter code for this part is contained in the folder bin_detection. You must implement one of the models from Part 1., either Logistic Regression or Naïve Bayes or Gaussian Discriminant Analysis for color classification. Similarly, your report must discuss the model you implement and the results it obtains on both the pixel classification and recycling bin detection tasks. Once you are done with the required model, you are free to (but it is completely optional and will not receive extra credit) try other machine learning approaches if you have time, e.g., decision trees, support vector machines, neural networks etc.

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html
[2] https://natanaso.github.io/ece276a/ref/ECE276A_1_Introduction.pdf

Figure 1: This project focuses on blue color classification and detection of a blue recycling bin based on its rectangular shape

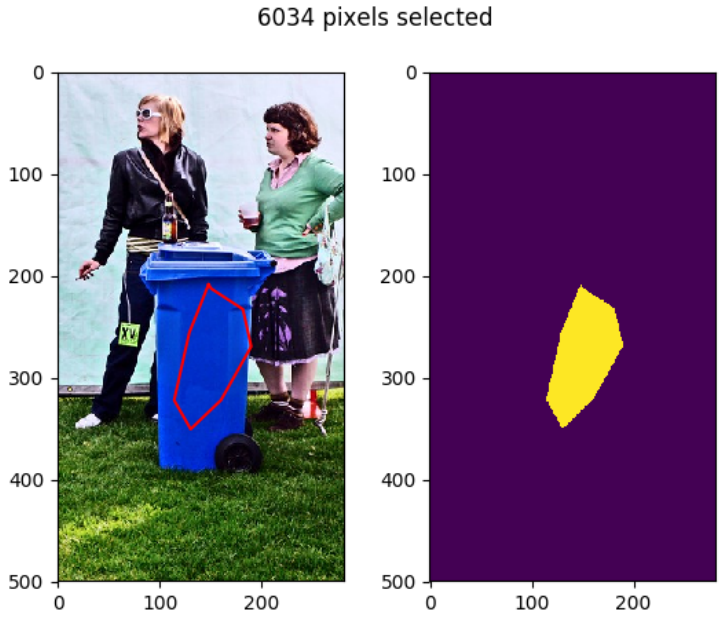

Figure 2: Use the **roipoly** labeling tool to generate color model training data.

- Unlike Part 1., color training data is not readily available here. Hand-label appropriate regions (polygonal sets of pixels) in the training images with discrete color labels as shown in Fig. 2, using the provided **roipoly** function[3]. We are especially interested in the recycling bin blue surface (positive examples) and regions containing similar-colored areas that are not a blue recycling bin (negative examples). Lighting invariance will be an issue, so you should consider what is an appropraite color space to use. Note that we are labeling recycling-bin-blue regions, not recycling bins. You should build a color classifier at least for recycling-bin-blue but using additional color classifiers (e.g., brown or dark green) might improve the performance.
- You *must* implement and present results from a classification approach discussed in class (Logistic Regression or a Single Gaussian Generative Model). Your model must generalize to classifying pixels in new unseen images. To prevent overfitting on the pixels from the training images, train your algorithms using the training set and evaluate their performance on the validation set. This will allow you to compare different parameters for the probabilistic models and different color space choices.

[3] https://github.com/jdoepfert/roipoly.py

- Once the color regions are identified, you can use shape statistics and other higher-level features to decide if there are any recycling bins and where they are located in the image. Try all possible combinations of (sufficiently large) blue regions and compute a recycling-bin shape "similarity" score for each one. Identify the coordinates of a bounding box for the regions with high "similarity" score. The **regionprops** function from the scikit-image package[4] will be particularly useful here. You can also use morphological operations[5] (e.g., dilation or erosion) and the standard dimensions of a recycling bin, shown in Fig. 1, in case this information is useful for detection. Your algorithm should be able to quickly classify and display both color classification and bin detection results on a new set of test images.
- You should use the provided starter code "bin_detector.py" and implement the two functions "segment_image()" and "get_bounding_boxes()". For this file, please do not change the file name, class name, function names, or function arguments because they will be used by the Gradescope autograder. Also, **do not use any built-in functions that implement a core part of this project** (Logistic Regression, Naïve Bayes, Gaussian Mixtures, EM). If you are not sure, then ask the TAs if a particular package may be used.

3. Write a project report describing your approach to color classification and blue recycling bin detection. Your report should include the following sections:
   - [5 pts] **Introduction**: discuss why the problem is important and present a brief high-level overview of your approach
   - [10 pts] **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in using precise mathematical terms.
   - [20 pts] **Technical Approach**: describe your approach to color classification and recycling bin detection.
   - [10 pts] **Results**: present your training, validation, and test results, and discuss them – what worked, what did not, and why. Make sure your results include (a) the final parameters used by your color classification models (e.g., the mean and covariance for Naïve Bayes or the weights for Logistic Regression), (b) examples of segmented color images showing the classified blue regions in Part. 2, and (c) the bounding box coordinates that your algorithm detects on each validation image. You should include discussion of your results and any other qualitative or quantitative results that will help present your work.

[4] https://scikit-image.org/docs/dev/api/skimage.measure.html
[5] https://scikit-image.org/docs/dev/auto_examples/applications/plot_morphology.html