Naïve Traffic Light Detection

# EE/CNS/CS 148B – HW1

## What algorithms did you try?

1. **“Catch all”:** Draw a single bounding box enclosing the entirety of the top-half of the image.
2. **“Find red”:** Iterate pixel by pixel. If the red channel is above a minimum threshold, and the green and blue values are within a predetermined range, enclose the surrounding area by a bounding box.
   1. Hyperparameter tuning:

## How did you evaluate algorithm performance? Can you think of any situations where this evaluation would give misleading results?

* **Precision:** out of all the bounding boxes outputted by my model, what percentage correspond to a red light.
  + An overly cautious model, that makes very few predictions trying hard for all to be correct, might not make a lot of mistakes (so high precision) but might miss identifying a lot of red traffic lights (low recall).
* **Recall:** what percentage of all red traffic lights in the image were correctly identified by my model.
  + We can “fool” this metric if we simply predict a single bounding box enclosing the entire image! Then we are guaranteed to find all red traffic lights, as they would be somewhere in the bounding box.
* **Amount of wasted space in bounding box:** Although you can find all traffic lights by boxing the entire image, most of the bounding box would be “wasted”. A good model would output bounding boxes sufficiently big to enclose the entire traffic light but no bigger than that.

## Which algorithm performed the best?

“Find red” was the clear winner. It had the best performance overall when considering the three metrics. It had high recall because it successfully identifies most red traffic lights. Its precision was not terrible, but it was certainly brought down by a lot of false negatives since there are objects in the street that have a red color similar to traffic lights that are not actually traffic lights. Also, its predicted bounding boxes were not big and wasteful.

## Identify a potential problem with your approach and propose a solution.

One of the biggest issues is that my model predicts a fixed-size bounding box, which was a hand-picked hyperparameter chosen so that in most images the bounding box seemed reasonable sized. This takes advantage of the fact that Dr. Perona put together this dataset by taking pictures sitting in his car, so the distance and angle from which we see traffic lights is fairly consistent. If someone were to put a stair on the street and take a picture of a traffic light from very close, my model would catastrophically fail to predict an appropriate bounding box. One potential solution would be to increase the size of the bounding box until it no longer encloses red and just add some additional margin to enclose the frame of the light, but this would be very annoying code to write and would be overkill for this dataset, anyway.

## Good predictions examples

|  |  |
| --- | --- |
| Graphical user interface, application  Description automatically generated | A picture containing text, sky, outdoor, light  Description automatically generated |
| A picture containing text, sky, outdoor, screenshot  Description automatically generated | A picture containing calendar  Description automatically generated |

1 The model was able to achieve perfect precision and recall on multiple instances. On the first three images, it found all red lights without making additional incorrect predictions. On the bottom right image, it was not fooled into predicting a red light when there wasn’t any.

## Bad predictions examples

|  |
| --- |
| A picture containing text, tree, outdoor, scene  Description automatically generated |
| Graphical user interface  Description automatically generated |

Hard images where often those with car taillights, which were often mistaken for traffic lights. Other times, traffic lights were not identified (top image), since lighting conditions made the shade of red in some traffic lights be different than that of the traffic lights used to fix the ranges of red (hyperparameters) that the model selected for.