EE/CNS/CS 148 HW 2: Red Light Detection

Johanna Karras

April 15, 2021

Assignment Description: In this assignment, I develop a simple matched filtering algorithm for detecting red lights in urban scenes using *only NumPy functionality* and evaluate the model using IoU and precision/recall curves.

GitHub Repo:
`https://github.com/JSKarras/EE148-Detecting-Red-Traffic-Lights-pt2`

1 Matched Filtering Red Light Detection Algorithm:

1. Normalize each input image `I` with respect to the image kernel (a red light square) `T`:

$$I_p[:,:,0] = (I_p[:,:,0] - \hat{T_r})/255$$
$$I_p[:,:,1] = (I_p[:,:,1] - \hat{T_g})/255$$
$$I_p[:,:,2] = (I_p[:,:,2] - \hat{T_b})/255$$

   where $\hat{T_r} = \sum T[:,:,0]/|T|$, $\hat{T_g} = \sum T[:,:,1]/|T|$, and $\hat{T_b} = \sum T[:,:,2]/|T|$, or the mean of each color channel of the kernel.

2. Pad the input image with `T_size - 1` 0's on the left, right, top, and bottom, where `T_size` is the kernel image size, in order to preserve image size.

3. For the image kernel `T`, iterate over kernel sizes `t_size` = [5, 3, 2]...

   - Iterate over image patches `I_p` of the `t_size`, taking `stride` pixels (default 1) in between each image patch.
   - For each image patch, compute the convolution `conv` between `I_p` and `T`: `conv` = `I_p` $*$ `T`.
   - Store the value of each convolution in a heatmap.

4. Average the values of the heatmaps into a final heatmap.

5. Iterate over the heatmap and for pixels with `conv` >= `threshold = 0.85*max(heatmap)`, store the pixel as the upper-left and compute the bottom-right coordinates 8 pixels away, (ul_x, ul_y, br_x, br_y), but only if...

   - The image patch is at least 30 pixels away (Euclidean distance) from any other previously stored image patch.
   - The normalized flux of the patch is between 0.9*`T_flux` and 1.1*`T_flux`, where `T_flux` is the flux of the kernel.

   These are the bounding box predictions for red lights.

6. Overlay the stored image patches as bounding boxes over the original image.

7. For each input image, compute the IoU (intersection-over-union) of each predicted image and ground truth label: $\frac{\texttt{Area of Intersection}}{\texttt{Area of Union}}$. Then,

   - Count images whose IoU and confidence (r-value) are greater than the threshold values as "True Positives".
   - False Positives = Total Predictions - True Positives
   - False Negatives = Total ground truth objects - True Positives

8. Compute precision and recall. Plot precision vs. recall.

$$\texttt{precision} = \frac{TP}{TP + FP}$$
$$\texttt{recall} = \frac{TP}{TP + FN}$$

2 Template and Predicted Heatmap:

For each template size, I iterate over all patches of the input image that match the size of the template, taking `stride` (default 1) pixels between each image patch. I calculate the convolution between the image patch and template and store them in a temporary array. Once I have done so for each size of template, I take their pixel-wise average to produce the heatmap. Then, pixels whose average heatmap score is above the threshold represent the center of red-lights. To compute boxes, I take these center coordinates and subtract 5 pixels to get the upper-left coordinates and add 5 pixels to get the bottom-right coordinates of the bounding box.

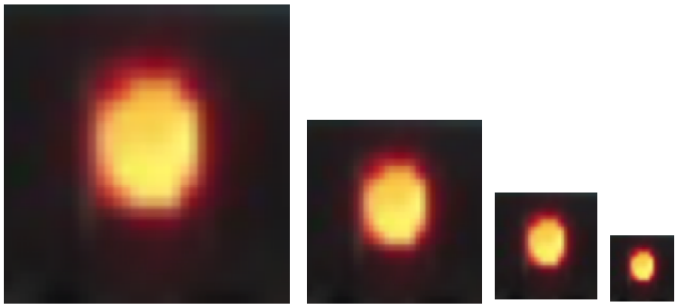
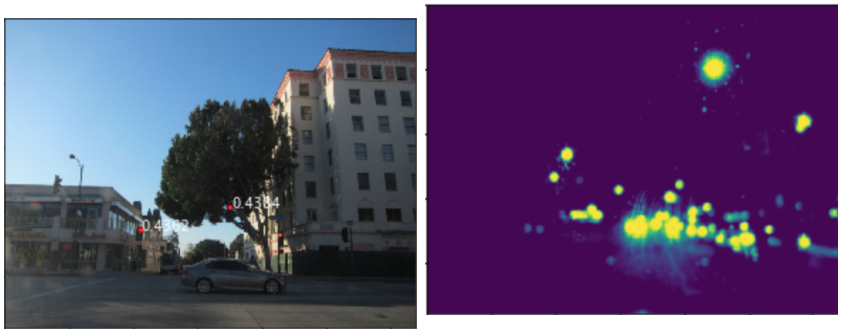


Figure 1: The chosen image template of a red traffic light rescaled to different sizes.



(a) Sample image with red light detections based on heatmap.

(b) Average heatmap generated from image (left) and template images in Figure 1.

Figure 2: Image and heatmap example.

3 Example "Easy" Images:

(a) This is an easy image, because there aren't other red dots that could be mistaken for traffic lights.

(b) This image is easy, because there are no red lights or any other red objects to confuse the model.

(c) This image is easy, because the red light is very visible and close to the choice of kernel in terms of coloring.

(d) This image is easy, because there are no deceptive objects and the traffic lights are very red and similar to the kernel.

Figure 3: Examples of images with easy-to-detect red traffic lights.

4 Example "Hard" Images:

(a) This image is hard, because the bus tail light looks like a red traffic light.

(b) This image is hard, because the red light is very far away and very small.

(c) This image is hard, because the rain and darkness blur and warp the lights white lights appear orange/red.

(d) This image is hard, because there are a lot of reflected red lights and red tail lights.

Figure 4: Examples of images with hard-to-detect red traffic lights.

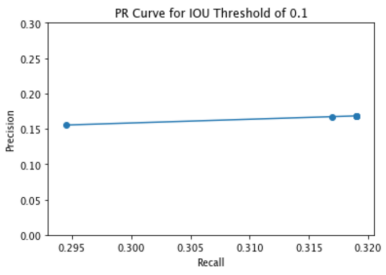
5 Original Algorithm Results:

In this section, I analyze the precision-recall curves of the original algorithm's predictions at different IoU (intersection-over-union) thresholds for both training and testing datasets.

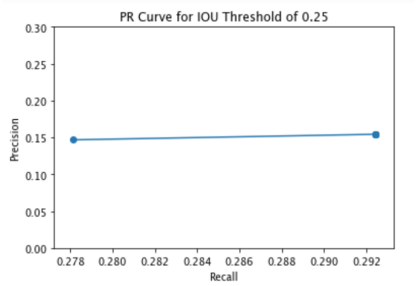
As I increase the IoU thresholds, fewer predictions are classified as "true positives" and more are identified as "false positives." As a result, there is lower precision and recall with a higher IoU thresholds. Additionally, more predictions pass the confidence threshold with lower IoU threshold. For example, compare the curves where the IoU threshold is 0.1 versus 0.75. In the former, there is a non-zero number of predictions that are labelled as true positive for multiple confidence thresholds, but in the latter, only in one confidence threshold (0.0) allows for any predictions to pass as true positives. Further, the in Figure 5a, the highest precision is 0.17 and the highest recall is 0.32; whereas in Figure 5d, the highest precision is 0.015 and the highest recall is 0.30675. This makes sense, since precision is a measure of how many predictions are true positives and recall is a measure of true positives over all ground truth values.

There is not much difference between the training and test curves. This makes sense since the image templates were tuned to the training images, which were similar to the testing images.

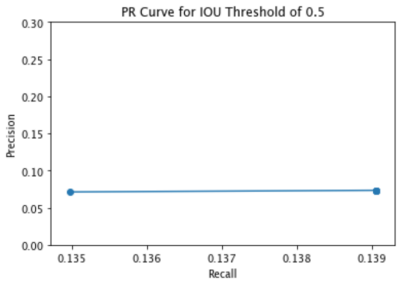
5.1 Training Set:



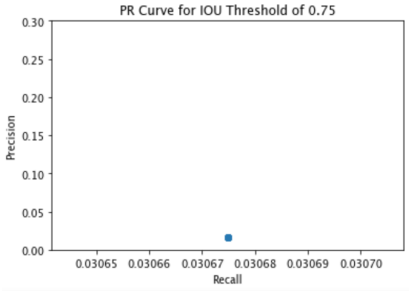
(a) PR-curve for IoU threshold = 0.1.



(b) PR-curve for IoU threshold = 0.25.



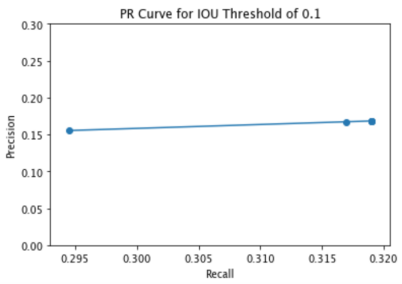
(c) PR-curve for IoU threshold = 0.5.



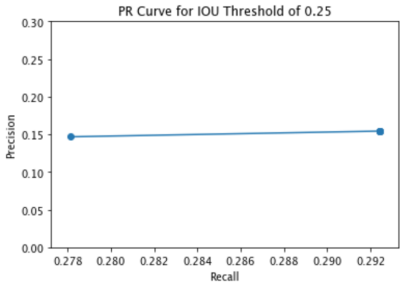
(d) PR-curve for IoU threshold = 0.75.

Figure 5: Precision-recall curves for specific IoU thresholds, varying confidence thresholds from 0 to 0.5, on the training dataset.

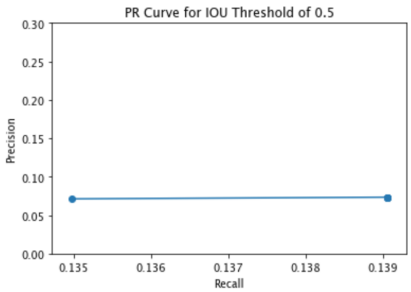
5.2 Test Set:



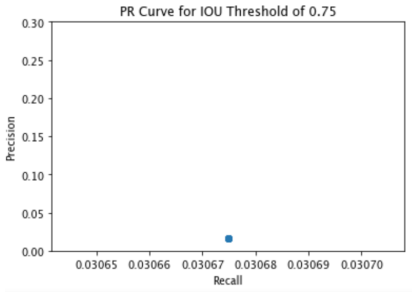
(a) PR-curve for IoU threshold = 0.1.



(b) PR-curve for IoU threshold = 0.25.



(c) PR-curve for IoU threshold = 0.5.



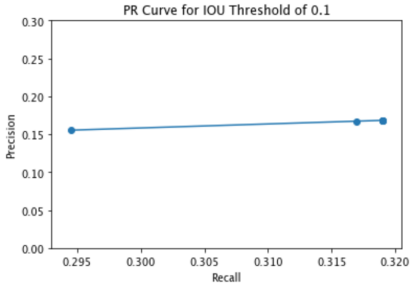
(d) PR-curve for IoU threshold = 0.75.

Figure 6: Precision-recall curves for specific IoU thresholds, varying confidence thresholds from 0 to 0.5, on the test dataset.

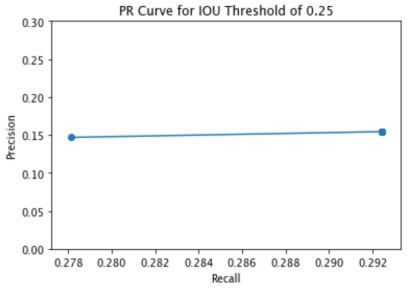
6 "Weakened" Algorithm Results:

Once I remove an optimization from my original algorithm, that is when I use only one size of template and do not average over multiple heatmaps, the model performance does not significantly for training or testing dataset. Comparing my original algorithm to the weakened version shows the relative importance of the optimization that I performed. For instance, it shows that the averaging of heatmaps did not help my particular algorithm very much. This is surprising, because the averaged heatmaps are easier to read by eye. Perhaps it the other parameters that tune out false positives also reduce the benefits of averaging. However, in other contexts, comparing the original algorithm to a weakened version with "hacks" taken out could reduce overfitting to the training set, or at least show which tricks are specific to the training set.

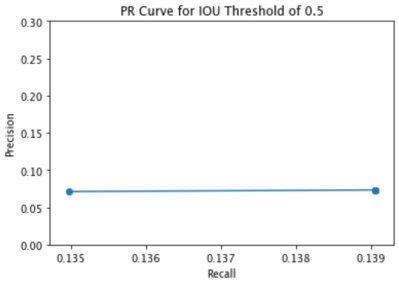
6.1 Training Set:



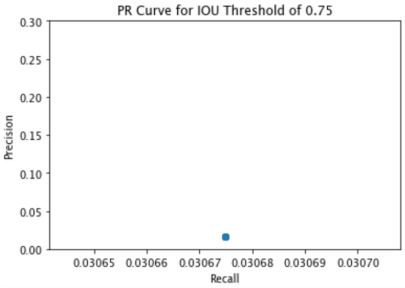
(a) PR-curve for IoU threshold = 0.1.



(b) PR-curve for IoU threshold = 0.25.



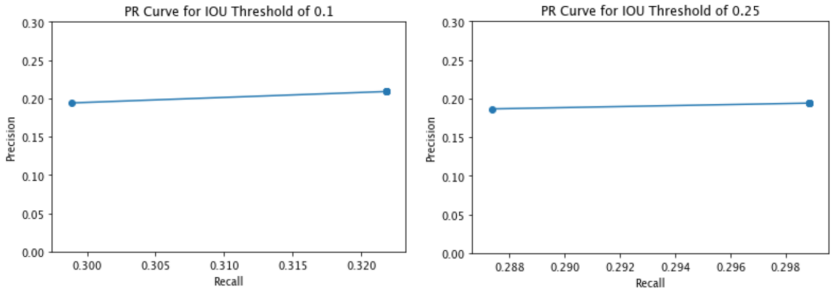
(c) PR-curve for IoU threshold = 0.5.



(d) PR-curve for IoU threshold = 0.75.

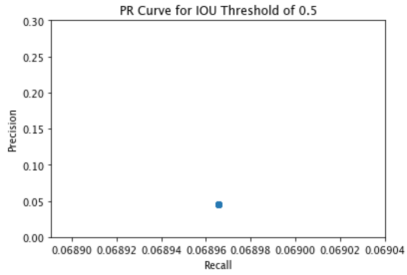
Figure 7: Precision-recall curves for specific IoU thresholds, varying confidence thresholds from 0 to 0.5, on the training dataset using a weakened version of the algorithm.

6.2 Test Set:

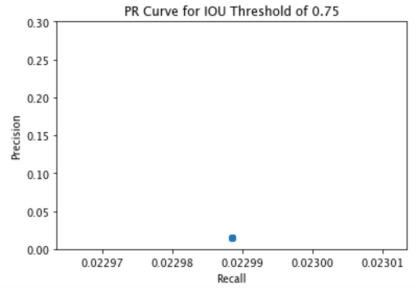


(a) PR-curve for IoU threshold = 0.1.

(b) PR-curve for IoU threshold = 0.25.



(c) PR-curve for IoU threshold = 0.5.



(d) PR-curve for IoU threshold = 0.75.

Figure 8: Precision-recall curves for specific IoU thresholds, varying confidence thresholds from 0 to 0.5, on the test dataset using a weakened version of the algorithm.

The code to generate these curves can be found at:

GitHub Repo:
`https://github.com/JSKarras/EE148-Detecting-Red-Traffic-Lights-pt2/hw02.ipynb`