UDACITY

‹ Return to "Intro to Self-Driving Cars" in the classroom

Traffic Light Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Well done! You have completed the Traffic Light Classifier project! From the effort you put into this project, I'm sure you will do well in SDCND!

This is the last project in ISDC. All the best in your future studies!

Please see the detailed feedback below.

Below is a link of a Numpy tutorial that might be helpful to you: http://cs231n.github.io/python-numpy-tutorial/ (also, CS231 refers to a series of lectures on neural networks that you can find on Youtube by searching that name)

Also, for this project (and the previous ones), you have access to mentors in the Study Hall (link on the left side of the Classroom page). Inside Study Hall, there are rooms for each of the projects. If you find you need help with a project and the mentors for that room are non-responsive, please send a message to mentorship-support@udacity.com so that they can help correct this issue (this applies to SDCND as well, if you decide to sign up for that ND as well). All the best in your future studies!

Notebook Questions

In the project notebook, all questions are answered. (There are two questions total.)

Good description of the features you used. Consider using the H channel for finding specific colors as an additional feature that can increase the robustness of your classifier. Also, consider using the positional information of the brightest pixels to increase the accuracy of your classifier.

Great analysis of the weaknesses of your classifier! You are correct that images that are difficult for a human to classify would also be difficult for a computer (although that is mostly a question of how difficult it is to set the correct parameters for a classification model). How could you adapt your classifier based on these weaknesses?

Pre-processing

All input images (before they are classified) should be processed so that they are the same size.

You standardized all the images by resizing. This would also be the place to do any cropping (to avoid processing extra data unnecessarily) and adjustments to the lighting in the images (histogram equalization comes to mind).

All labels should be a one-hot encoded vector of length 3. Ex. 'yellow' becomes: [0, 1, 0].

Your one-hot encode function converts all labels to length 3 vectors, nicely done! What does it do for invalid labels (not red, yellow, or green)? Consider raising a specific error for invalid labels so that the user can correct the situation. For this dataset it isn't a problem, but it is good coding practice to take note of this for future datasets you may work with in other projects (especially in a professional setting).

Create a brightness feature

Using HSV colorspace, extract a feature from a traffic light image that represents the level(s) of brightness in an image. This feature can help classify any traffic light image. A feature can be a list, array, or a single value.

I see that for your features, you masked and cropped the images using the HSV colorspace and then got the sum for each of the R, G, and B color channels. As you mentioned though, the yellow color is difficult to lock on to in this colorspace. Consider instead using the H channel from the HSV colorspace to pick out exact colors that you want.

This tutorial shows you how to do this with Python specifically: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html Note that in OpenCV, the H channel ranges from 0 to 180 in values, which is important as you will need to scale down if you use values from other places. This website will help you have a good sense of what values of H represent which colors: http://colorizer.org/ The H channel here ranges from 0 to 360, so dividing values by 2 will allow you to use them in OpenCV.

Another option would be to use the S or V channels to find the position of the brightest pixels. By comparing the indices of the brightest pixels, you can decide whether they belong to the top (red light), middle (yellow light), or bottom (green light).

Classification Model

Using any created features, write a classification function that takes in a standardized RGB image and outputs whether a traffic light is red, yellow, or green as a one-hot encoded label.

I see that your classification model uses the R, G and B color sums and compares them to see which is the largest to return your prediction vector. This is a good approach to a creating a classification model.

Model Evaluation

The model must have greater than 90% accuracy on the given test set.

Your classifier achieved 91.6% accuracy on the test set, well done! You could try taking pictures of traffic lights in your own neighborhood and testing your algorithm on those as a next step, but I would recommend trying to achieve 95% first.

In the given test set, red traffic lights can never be mistakenly labeled as green.

No red traffic lights were mistakenly labeled as green! This is a great basic safety feature of your classifier!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review