Interview Question:

Paul Estano
Red Round Sign Detection

February 20, 2019

1 Dataset

The full data set contains 900 images with 43 different sorts of traffic signs. Metadata are stored in a csv file called "gt.txt". The first cell is the image filename, the 2nd and the 3rd cells corresponds to the coordinates of the higher left corner of the bounding box, the 4th and the 5th cells corresponds to the coordinates of the lower right corner of the bounding box. I use pandas to load the whole file as a DataFrame.

This 43 different signs can be divided in 3 main categories : prohibitive, danger, mandatory. The sign we want to detect (red circle) corresponds to the prohibitory category.

This prohibitory category includes 13 different traffic signs. Once we extract all of the images containing at least one prohibitory traffic sign, we only have 392 images left.

Hence, our dataset is not big enough to train a neural network. The common solution to this problem is called *data augmentation*.

1.1 Data Augmentation

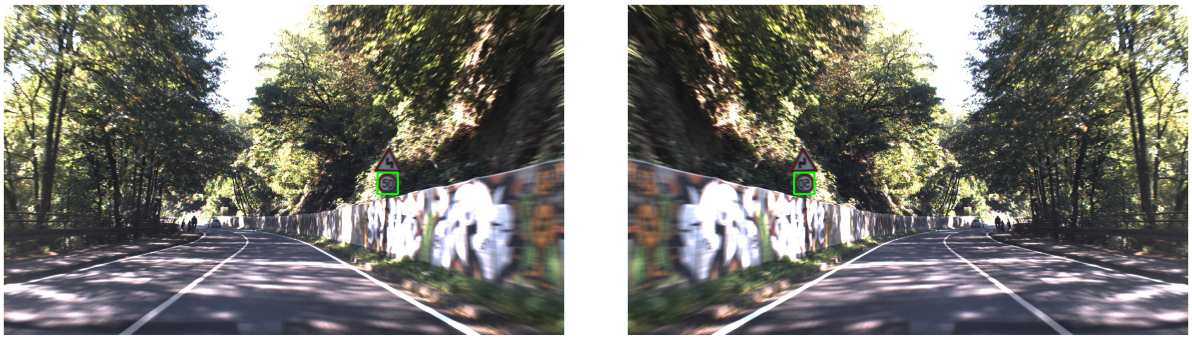
Data augmentation here means we are going to derive new images from the base dataset. There are several ways of doing this. Here I will focus on the 4 most commonly used techniques : flipping, rotation, translation and scaling. This 4 techniques rely on modifying geometric parameters of the base dataset images.

1.1.1 Flipping

The idea is to flip the images horizontally or vertically. Here, flipping the images vertically would not make any sense since we don't need to detect vertically flipped images of traffic sign. However, horizontally flipping all the images of the dataset enlarge our dataset by a

factor 2. To flip the image we use the opencv function *cv2.flip*. Then we have to compute the coordinates of the new bounding box:

$x' = width - x$ and $y' = y$



(a) Original Image (b) Flipped image

Figure 1: Flipping

1.1.2 Rotation

This technique is similar to the first technique. Here, we rotate all the images by +/- 10.

The problem with this technique is that we loose some data which is placed beyond image boundaries. To counter this effect we need some interpolation in the regions where we don't have any data.

Lots of python libraries implement an image rotation function but only a few of them integrate an interpolation algorithm. Scikit-image *transform.rotate* function implements both the rotation and the interpolation. This feature allows us to generate images without a black background or without resizing them. I choose a reflect interpolation in order to keep some continuity in the background. It works well for pictures in natural environment such as the following pictures.

Computing the bounding box coordinates of the rotation image is a bit trickier than what I did for the flipped image. Indeed, I used the 2D rotation matrix to implement this :

$x' = x\cos(\theta) - y\sin(\theta)$

$y' = x\sin(\theta) + y\cos(\theta)$

Furthermore, the algorithm has to check that the rotation does not move one of the image traffic signs beyond the image boundaries. If it does, we drop the transformation.

Figure 2: 10 rotated image

1.1.3 Scaling

Here we just rescale the images of the dataset. Since the images are already relatively big (1360 x 800 pixels), I downsample with a factor of 0.5 and upsample with a factor of 1.2.

To compute the coordinates of the new bounding box we only have to multiply the old ones by the scale factor :

$x' = x \times \alpha$ and $y' = y \times \alpha$ where α is the scale factor.

1.1.4 Translation

The idea is to vertically and horizontally translate the original image. Here we randomly translate the image on the Y and the X axis. The padding values are computed with the function *numpy.random.randint* based on ranges provided by the user. To compute the new coordinates of the bounding box we only have to subtract the padding value to the original coordinates:

$x' = x - padding$ and $y' = y - padding$

1.1.5 Noise

The idea is to add some noise to the original image.

Here I use a gaussian noise but one could also use some other type of noise (for instance salt and pepper).

Figure 3: Translated image

Figure 4: image with gaussian noise

1.1.6 Improvements

The methods above only allow to modify some of the geometric parameters of the images but can't modify the whole environment. For that matter, one of the bad aspect of this dataset is that the majority of the pictures (if not every picture) has been taken in daylight without snow or rain while we would like to detect the traffic sign under every weather condition or lightning condition.

To enlarge a dataset using existing picture by changing the weather condition we would need to use more advanced techniques such as generative adversarial networks like this one :

`https://junyanz.github.io/CycleGAN/`

In order, to have more diversed environment and weather conditions in the pictures one could have also chosen an other base dataset like this one : `http://www.nlpr.ia.ac.cn/pal/trafficdata/detection.html`.

Indeed, this dataset contains more images (10000). Besides, its pictures have been collected under diversed weather conditions and lightness conditions.

2 References

`https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-`

`https://blog.paperspace.com/data-augmentation-for-object-detection-rotation-and-shearing`

`https://en.wikipedia.org/wiki/Rotation_matrix`