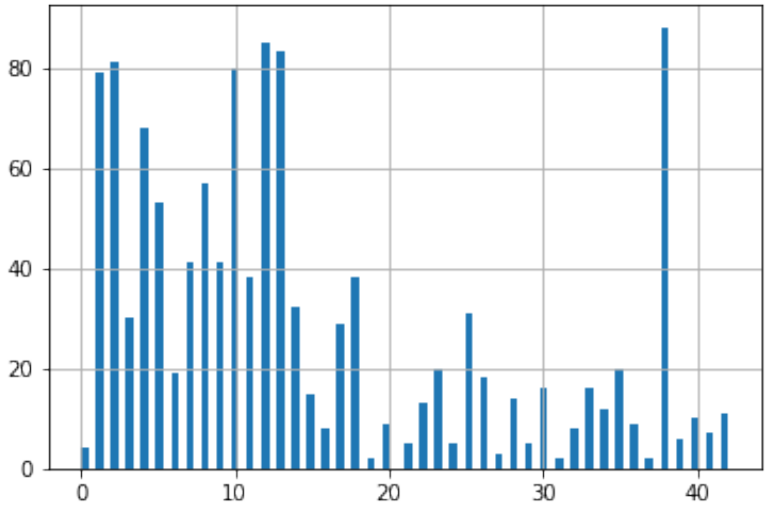
Report: Red Round Sign Detection

Part 1: Dataset:

The given dataset is the German Traffic Signs Detection Benchmark (GTSDB) dataset. The full dataset features:

- a single image detection problem
- 900 total images, each of 1360 x 800 pixels in PPM format
- the image sections that contain the traffic signs
- a file in CSV format containing ground truth labels

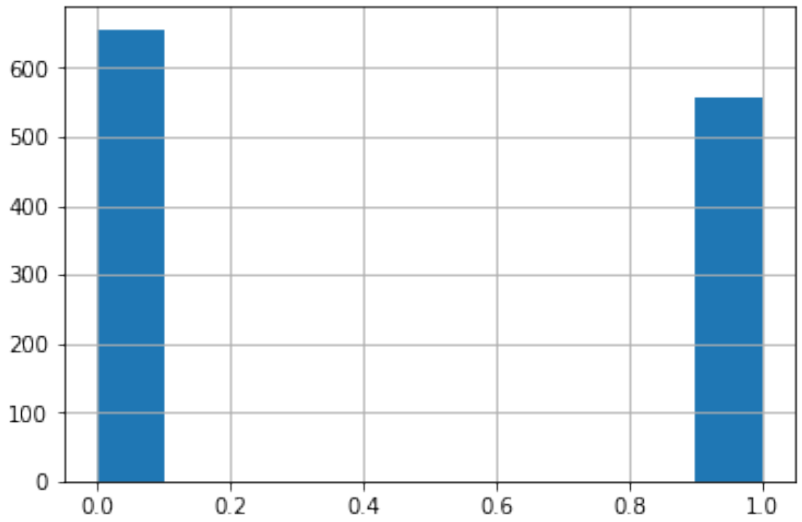
There are a total 1213 instances of different traffic signs across all 900 images, with uneven distribution of 43 classes, shown below:



The images are read from the PPM format, and converted to JPEG format, since it relies on Discrete Cosine Transform, which is extremely efficient in this lossy compression technique. The original dataset contains around 43 classes of different traffic signs, but our objective is to detect only red round signs in the images. The red round signs fall under the "prohibitory" class, and it corresponds to the following labels:

Red round signs = [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 16]

Therefore, I changed the ids of all images that contain red round traffic signs to 1, and make the rest of the ids 0. In that way, the problem can be alternatively formulated in a simple binary classification format, where the task to identify whether the traffic sign is a red round sign or not. This formulation makes the class distribution somewhat balanced, as shown below:



We have 557 instances of red round signs (class 1), and 656 are negatives (class 0).

Now, I randomly divide the entire dataset separately into train and test sets, keeping 20% of the data for test, and the rest for train and validation purposes. After this, we prepare our training and testing data in a format that allows us to feed into a neural net. For this purpose, we read each JPEG image, extract the traffic sign from it using the given bounding box coordinates, resize it to a fixed 48 x 48 x 3 format and perform contrast enhancement to reduce blurriness in the cropped images.

As a result, we save training and testing data in the format (970, 48, 48, 3) and (243, 48, 48, 3) respectively. We have 454 instances of red round signs in training data, along with 516 negative samples, making the training dataset well balanced. This allows us to use Accuracy metric for evaluation purposes, which is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = True positive, TN = True negative, FP = False positive and FN = False negative. Simultaneously, we convert all binary class ids into an one hot encoding format suitable for this task.

We notice that the total number of images is much less than that required in any standard deep learning model. One way to tackle this to perform Image Augmentation during training. We perform three types of augmentation using the open source "imgaug" library on every training batch of images fed to the model:

- Horizontally flip 50% of the images in the batch
- Vertically flip 20% of the images in the batch
- blur batch images with a Gaussian kernel with sigma that varies between 0 and 3.0

This partially alleviates the problem of having less data for training purposes, and prevents early overfitting. Another way of tackling problems with very less data is to use the knowledge of Transfer learning. According to this technique, models trained on one task capture relations in the data type and can easily be reused for different problems in the same domain. With transfer learning, we can take a pretrained model, which was trained on a large readily available dataset. Then try to find layers which output reusable features. We use the output of that layer as input features to train a much smaller network that requires a smaller number of data points.

Unlike this GTSDB dataset, there exists a much bigger and better dataset known as LISA Traffic Sign dataset, that consists of set of videos and annotated frames containing US traffic signs. It consists of 7855 annotations of traffic signs in 6610 frames, with varying sizes. More interestingly, images are obtained from different cameras, with varying size, position, occlusion and illumination. Another interesting dataset that we can explore is the CURE-TSR dataset, which consists more than two million traffic sign images that are captured in controlled challenging conditions. I believe that working on these type of datasets will really help understand how to effectively tackle diverse images and traffic signs and come up with better deep learning algorithms to solve the detection / classification problem.

In unconstrained environment conditions, object detection / recognition becomes a challenging task. The huge variation in object scale, orientation, category, and complex backgrounds, as well as the different camera sensors pose great challenges for current algorithms. Most importantly, object detection and tracking at night remain very important problems for visual surveillance. If scene is completely dark, then it might be necessary to use a thermal infrared camera, which can prove to be extremely costly. The images captured by traditional car cameras have low brightness, low contrast, low signal to noise ratio (SNR) and nearly no color information. One way we can handle this is by detecting the object using local contrast computed over the entire image. This can be done by measuring sub-image inter-frame differences. Then we can track the detected objects and remove falsely detected objects with the

help of a tracking algorithm, such as Kalman filter. Hand crafted feature extraction algorithms such as Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG) are partially invariant to image distortions and illumination changes, thus can be explored under dark environment conditions.

Also, there are many state-of-art deep learning algorithms such as RCNN, Fast RCNN, Faster RCNN and Mask RCNN which were introduced over the years to tackle such problems. The backbone of Faster RCNN and subsequent models is the Region Proposal Network (RPN), capable of generating region proposals for increased localization accuracy. Additionally, YOLO and SSD are two popular frameworks that can handle complex object detection tasks. YOLO architecture is more like a fully connected CNN, which splits the input image into multiple fixed sized grids, and for each grid generates 2 bounding boxes and class probabilities for those bounding boxes. Another separate technique involves feature pyramid networks (FPN) that extracts feature maps through a feature pyramid, thus facilitating object detection in different scales, at a marginal extra cost.