Name : Shubham Sidhwa

gtID : 903390326

AI-on-Smart-City-Infrastructure-Health-and-Safety-Condition-Assessment-and-Monitoring---Red-Round-Sign Detection

PHASE 2 : TRAINING :

Why did you choose this NN structure? What's the pros and cons of it?

I tried working with Convolutional neural networks and tried the YOLO algorithm for detecting red round signs in the given images.

The convolutional neural network consisted of 5 layers with the Relu activation function.

Layer 1: Convolutional. Input = 32x32x1. Output = 28x28x32
Pooling. Input = 28x28x32. Output = 14x14x32
Layer 2: Convolutional. Input = 14x14x32. Output = 10x10x64
Pooling. Input = 10x10x64. Output = 5x5x64
Flatten. Input = 5x5x64. Output = 1600
Layer 3: Fully Connected. Input = 1600. Output = 120
Layer 4: Fully Connected. Input = 120. Output = 84
Layer 5: Fully Connected. Input = 84. Output = 43

The layer 5 output of 43 corresponds to the 43 classes of signs that are given to us in the German Sign Dataset.

I tried working with this network and training and testing the dataset but was unable to complete it because the code used to throw unexpected errors every time. Also, I do not have a fast processing GPU hence my code used to hang every time I tried it.

This was my first time I worked on a problem concerning implementation of neural networks; hence it took me time initially to understand about different neural networks and their coding implementation by studying on the Internet about them.

I have prior experience of working with Image Processing Projects but not much with problems concerning large dataset problems.

Pros :

- The main advantage is their accuracy in image recognition problems.

Cons :

- High computational cost.
- If you don't have a good GPU they are quite slow to train (for complex tasks).
- They need a lot of training data.

How do you tweak your hyperparameters? What are they all about?

There are many hyper parameters to cNNs, such as the learning rate or kernel sizes to use.

Kernel sizes were adapted from to a size of 5x5 which is the de-facto standard kernel size for image classification.

What's your training strategy? How can you tell if it works?

For training, I used the tensorflow framework which allows to formulate a computational graph which is then heavily optimized by the internal tensorflow compiler.

Another advantage of tensorflow and similar frameworks, such as theano, torch or keras is the automated computation of derivatives.

What's the performance of your detection system? How do you evaluate it?

Due to unavailability of time and system constraints, I couldn't complete the neural networks part.

Hence, I tried this problem out without neural networks but just using individual images as input.

I first wrote a code for converting images from .ppm format to .png format as they are more easier and efficient to work with in image processing.

I observed that the entire dataset provided to us consisted of either red or blue color signs.

So, I extracted the blue and red color parts out of the image. Then, I found the contours in the image and found only my region of interest(ROI) in each image.

For this, all contours having area less than a certain minimum were ignored and all contours having area above a certain maximum were ignored.

What remained were either traffic signs or something related to that.

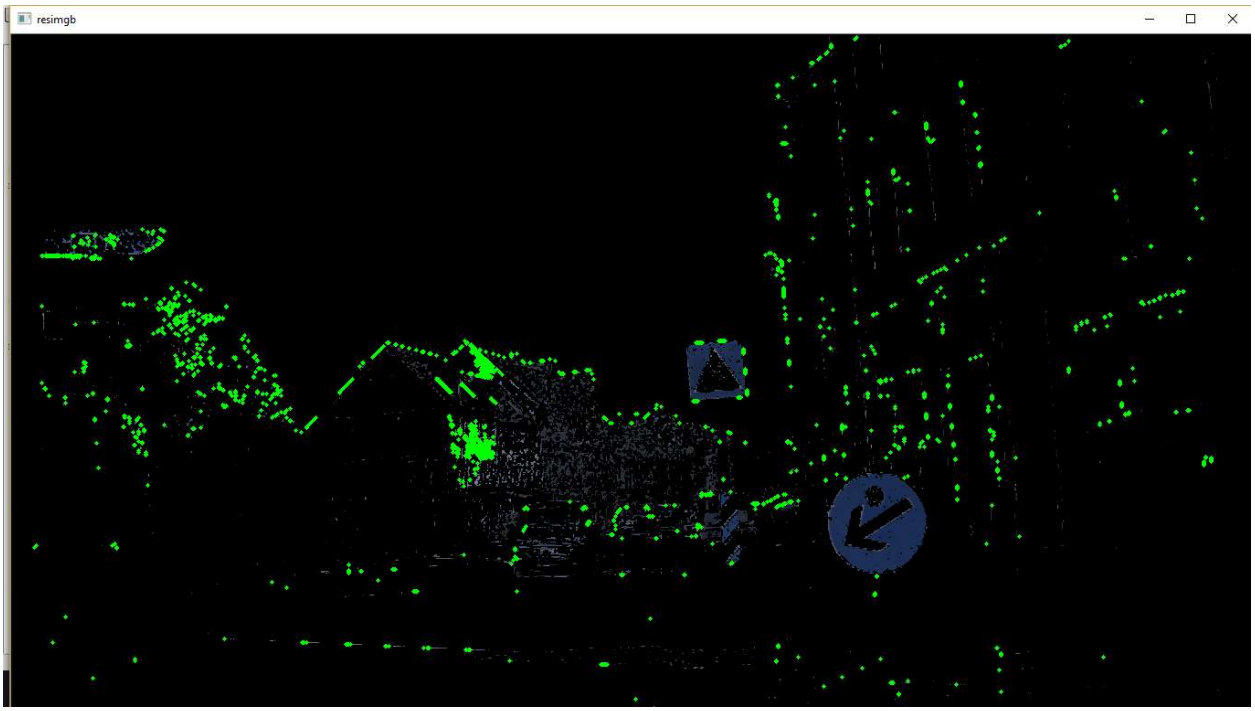
To then extract just the traffic signs, I decided to set another threshold above which are traffic signs.

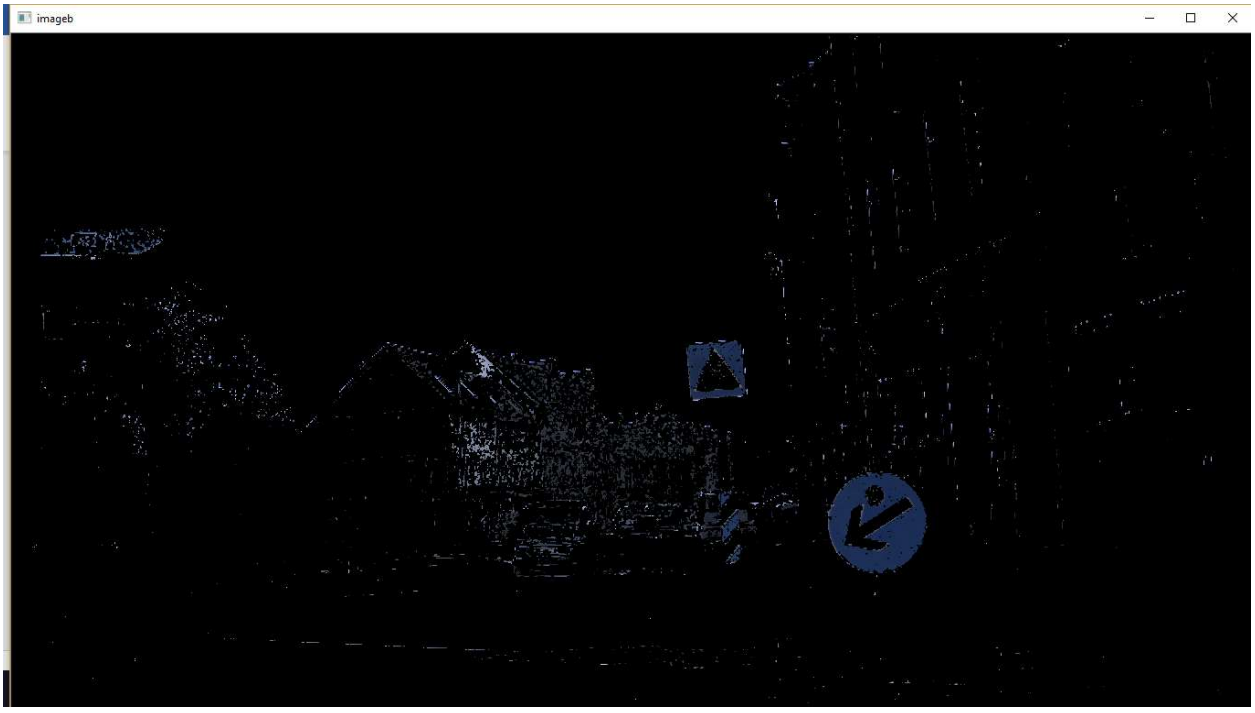
This has been set by trial and error.

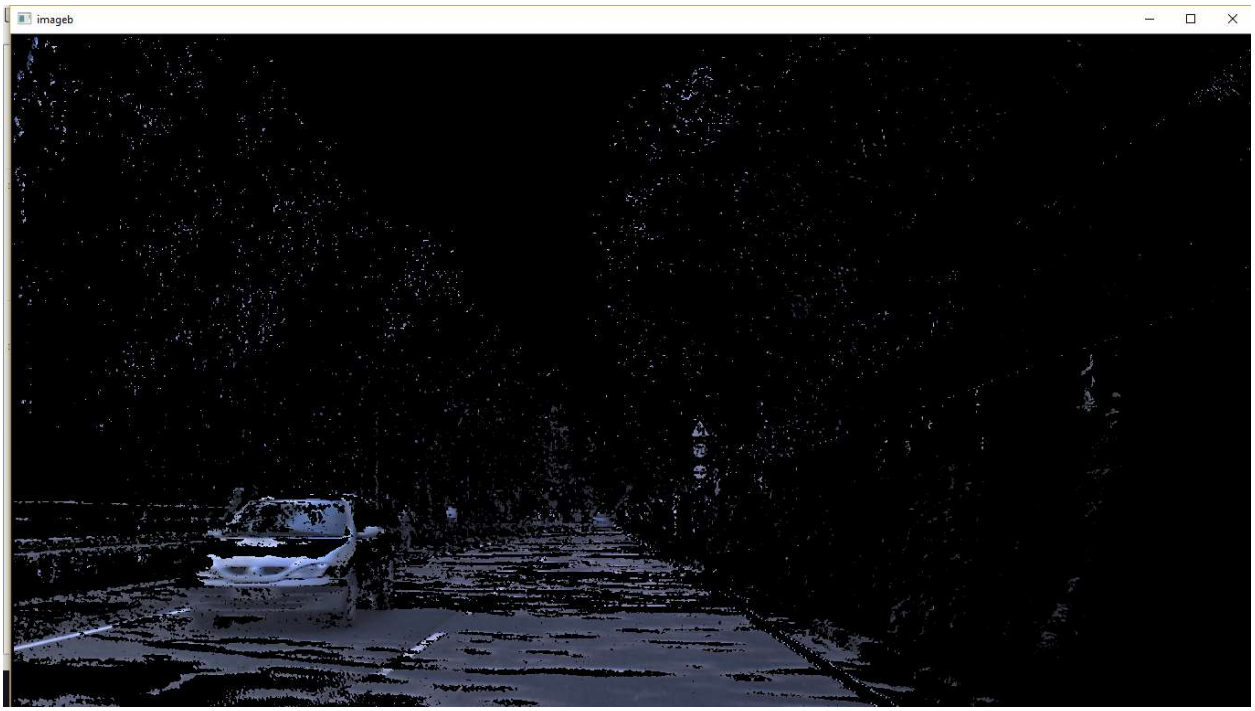
As a result, I tested this for the first 100 images and was successful for 53 images with an accuracy of 53%.

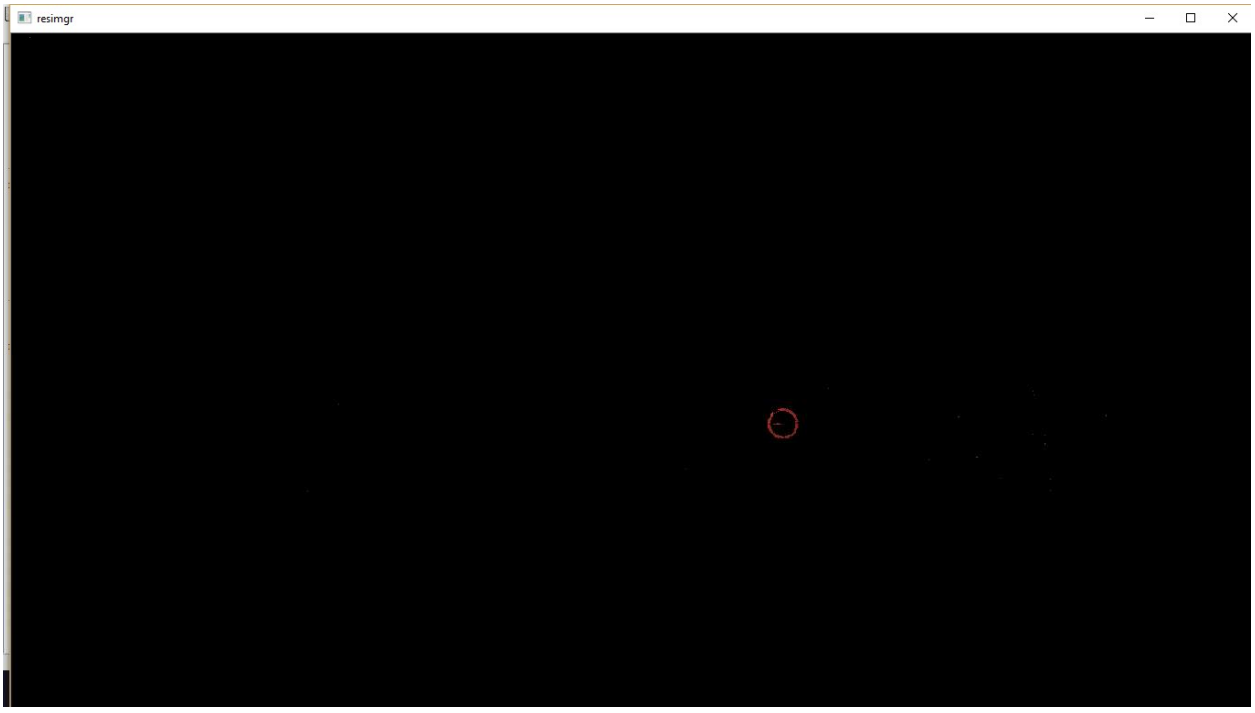
Output results :

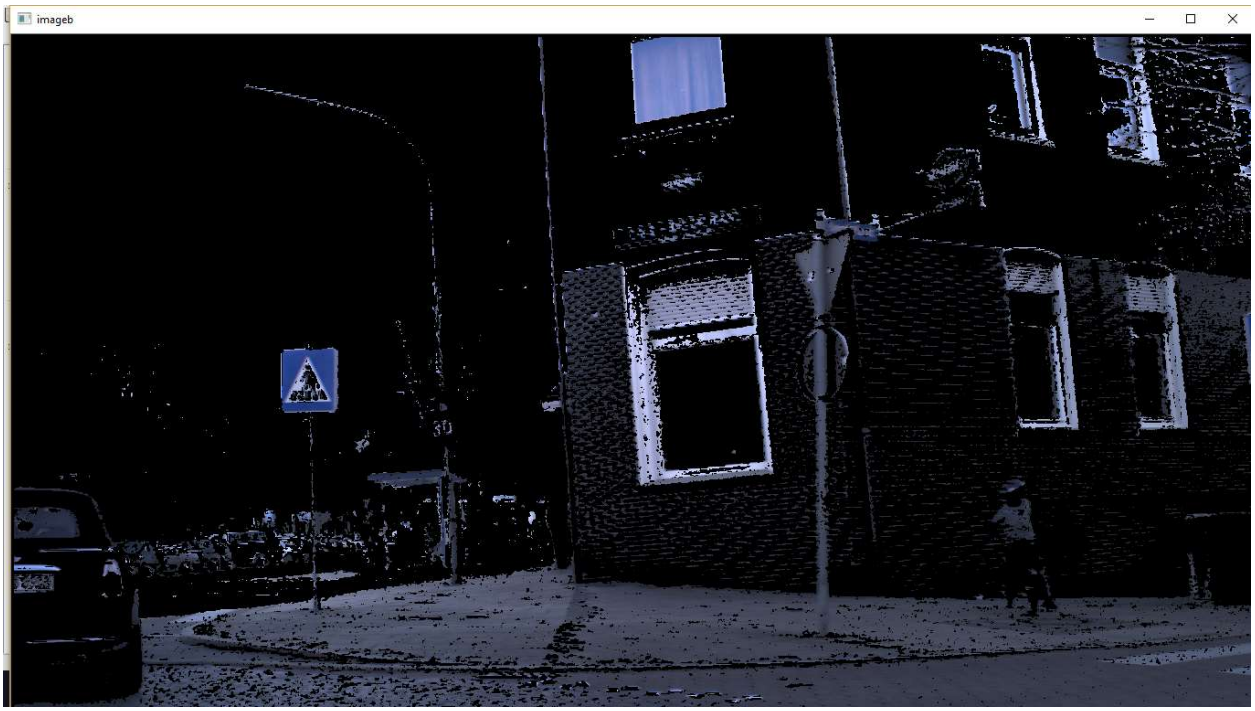

imageb


resimgb

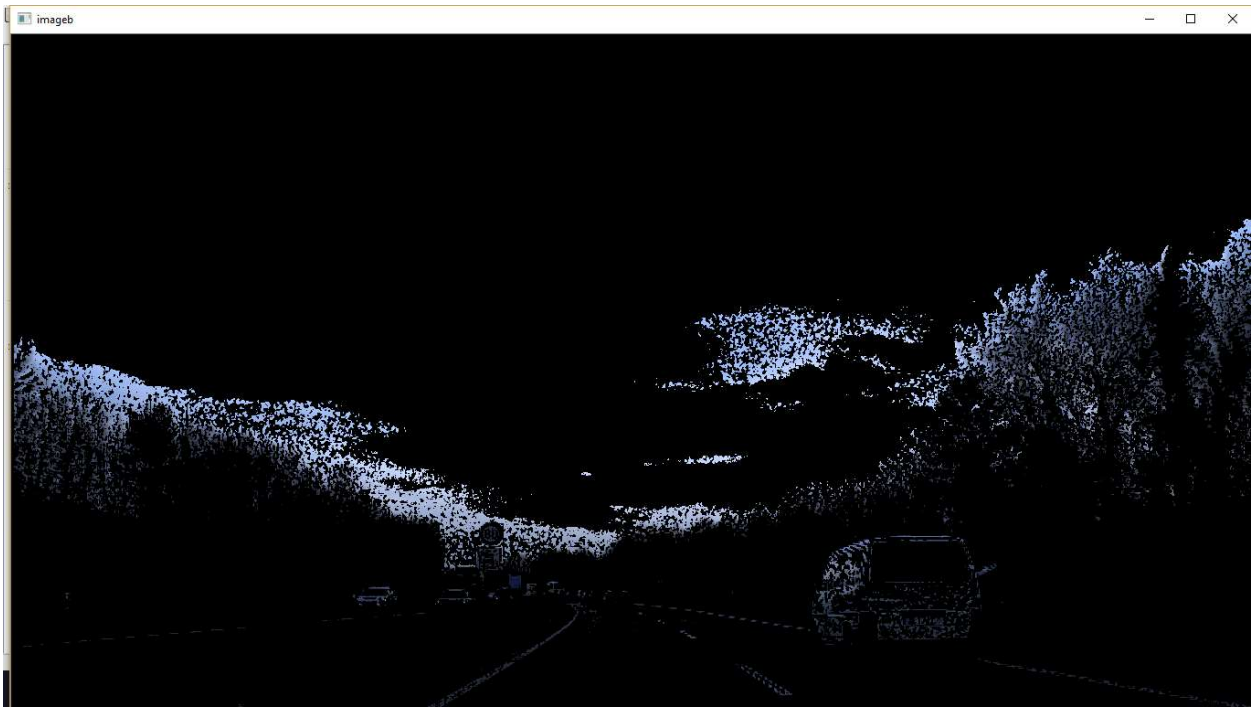

imageb


imageb


resimgr


imageb


imager


imageb