Road Sign Detection using Deep Learning: YOLOR

Younes EL BOUZEKRAOUI
Georgia Institute of Technology
ybouzekraoui3@gatech.edu

***Abstract*— This project is about image processing and object detection using neural networks , the purpose of this work is to prepare the data and train a deep neural network YOLOR on the he German Traffic Sign Detection Benchmark (GTSDB) to detect and classify road signs.**

I. INTRODUCTION

YOLO "You Only Look Once." is an algorithm that finds and recognizes different objects in an image in real-time using convolutional neural networks (CNN). It performs object detection as a regression problem and returns the class probabilities of the detected images. To identify objects, the approach requires just one forward propagation through a neural network, as the name implies.

There are multiple versions of YOLO v1-v5 , PP-YOLO.

In This Project we will Use the YOLOR version of YOLO

YOLOR is a state-of-the-art machine learning algorithm for object detection, different from YOLOv1-YOLOv5 in architecture, and model infrastructure. YOLOR stands for “You Only Learn One Representation”, not to be confused with YOLO “You Only Look Once”.

YOLOR is proposed as a “unified network to encode implicit knowledge and explicit knowledge together”.

II. DATA PREPARATION

A. *Exploring the Dataset*

Fig. 1. Random Images From the Dataset

The Dataset Used in this project is the German Traffic Sign Detection Benchmark (GTSDB). it includes 900 images of the size (1360 x 800 pixels) in PPM format , the annotations (Ground Truth) are stored in a txt file (gt.txt) that contains lines of the form:

ImgNo.ppm;leftCol;topRow;rightCol;bottomRow;ClassID

Example :

00052.ppm;833;352;870;389;36

Each line refers to an Object (ROI Region Of Interest) of an Image, There might be multiple objects in one Image.

There are 43 types of road signs. Their distribution is shown in the left histogram of Fig.1, we can see that the classes are imbalanced and there are some classes with fewer instances compared to others, this might affect the performance of the network when predicting those classes.

We grouped the 43 classes into 4 main categories:

- Prohibitory
- Mandatory
- Danger
- Others

Their distribution is shown on the right histogram of Fig.1, As we can see we have a reasonable number of instances in each category even if they are still not perfectly balanced.

```
Images with label Count : 741
Background images Count (Images Without Label) : 159
Total traffic sign instances in the dataset : 1213
Total classes in the dataset : 43
```

Fig. 2. Images and Classes Count

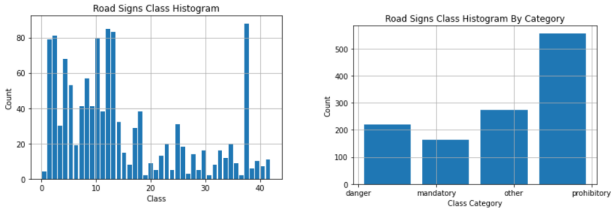

Fig. 3. Classes Distribution

In order to solve the Imbalanced Classes problem there many approaches that we can test:

- **DownSampling Approach** : Removing instances from the majority class to obtain a balanced data-set, however, with this approach, we are not using the total available data.
- **Weighted Class Approach** : Assigning different weights to the majority and minority classes. The

overall goal is to penalize the minority class for wrong classifications by increasing class weight while decreasing weight for the majority class.

- **Data Augmentation** : Creating similar images by changing (Hue , Colors , Orientation, Brightness ...) until the classes are Balanced.

For this Project, we will classify the road signs into the 4 categories cited above.

The images are transformed into .jpg format. We do not have to resize the images or change the aspect ratio because YOLOR already performs the resizing before training.

B. *Formatting the Labels*

YOLOR expects input in a given format, however, the annotations we have in the txt file (gt.txt) are in a different format, Thus we implemented a function "convert_to_yolor_annotations" to transform the labels to the expected format.

The figure below shows an example of the annotation format

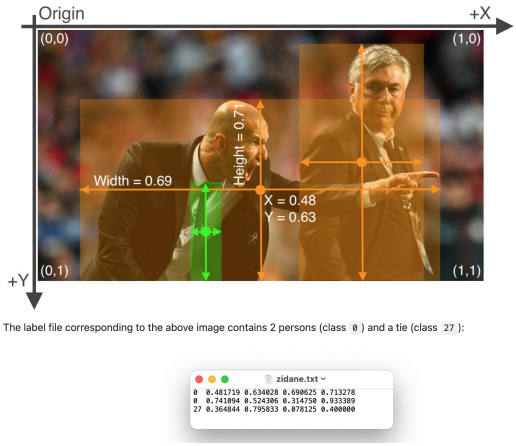

Fig. 4. Example : Labels in Yolo format

For each image we created a .txt file with the same name as the image, each line of this text file contains information about an object in the image.

Concerning Background Images , the images that does not contain any object, we create corresponding .txt file which is empty.

C. *Partitioning the Dataset*

The Data-set is divided into Train, Test, and Validation Subsets, 80% is used for training 10% for testing and 10% for validation.

The Data Repository is organized as shown in Fig.4

Two files (gtsdb.yaml and gtsdb.names) containing the paths to the images and the labels and Classes names are created and saved in the YOLOR repository (yolor/data)

Fig. 5. Repository Structure

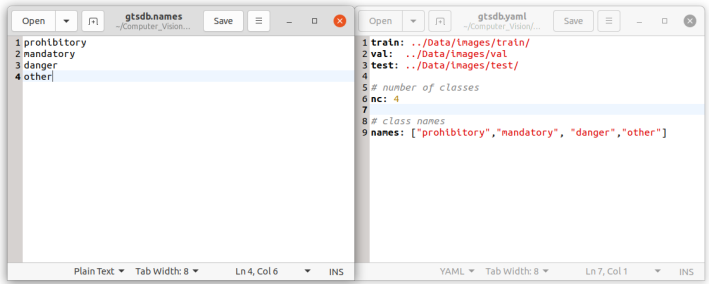

Fig. 6. gtsdb.yaml and gtsdb.names files

III. Discussion

- The Background Images (Images with no objects) are kept in the data set in order to help the network reduce the number of False Positives (Images with no objects classified as containing an object).

- When Training the Network We will test the different approaches we discussed earlier in order to deal with the imbalanced classes issue and test the performance of the network using each approach.

- We will give more details on the Data Augmentation when training the network and testing its performance.

References

[1] The German Traffic Sign Detection Benchmark `https://benchmark.ini.rub.de/gtsdb_dataset.html`

[2] You Only Learn One Representation: Unified Network for Multiple Tasks `https://arxiv.org/abs/2105.04206`

[3] YOLOR – You Only Learn One Representation (What's new, 2022) `https://viso.ai/deep-learning/yolor/`