Final Project Report

**Traffic Sign Detection**

August6th, 2023

**GROUP 3**

Thien An Trinh

Roman Burekhin

Athira Devan

Lester Azinge

# INTRODUCTION

We are living in a world that is moving towards automation. From robot arms assembling individual components into complete cars to smart household appliances that have been transforming our homes, the benefits of autonomous applications are undeniable. The automobile industry is following the same trend. Not only autopilot systems assist drivers by bringing them better driving experience, but they can also help reduce the number of accidents. For example, a vehicle with a smart traffic sign detection system can “see” all the signs ahead including those the driver could miss, and thereby perform proper actions timely in time-sensitive situation.

In that context, our project focused two paramount objectives: first, to meticulously benchmark a range of model architectures to identify the most optimal and efficient solution for this challenging task, and second, to deploy and demonstrate precise and reliable detection of critical traffic signs.

Before proceeding into further detail, it is crucial to address certain concepts related to how the task was framed. First of all, it was determined that the project is a computer vision task – a domain in which images are processed and analyzed in order to extract useful information that can drive decision-making (Arabnia et al., 2018; Yoshida, 2011). Second, within the computer vision domain, this project is specifically an *object detection* task where an image was analyzed not for obtaining the semantic meaning of the whole image (i.e., *image classification*), or for segmenting the image into meaningful regions (i.e., *image segmentation*), but rather to identify targeted objects that are present in the images and determine where on the images they are located. In this project, the objects of interest were *traffic lights, stop signs, speed limit signs* and *crosswalk signs*, which are the fundamental elements that guide drivers and traffic flow. Third, the algorithms required for this task were defined to be deep learning (DL) convolutional neural networks (CNNs) which has always been the state-of-the-art in the domain for a decade. Furthermore, the project also leveraged an advanced DL technique called *transfer learning*, in which neural networks that were pretrained on a large dataset are fine-tuned on the dataset of interest instead of being trained from scratch, and therefore are capable of attaining high evaluation scores in the new domain. The details of the data, pretrained-models, training and evaluation frameworks are now discussed in the following section.

# METHODOLOGY

## Data

### Data Source

The data for this project was from Kaggle and is available at this reference (*Road Sign Detection*). The dataset comprises of two folders, each of which consists of 877 files. The first folder is named “*images*” which contains 877 road sign images in PNG, whereas the other is named “*annotations*” and has 877 corresponding XML files that store the image annotations in the PASCAL VOC format (Everingham et al., 2015). The images belong to 4 distinct classes, namely *traffic light, stop, speed limit,* and *crosswalk* which are the target objects to detect (Figure 1, Figure 2).



Figure 1. Sample Images from the Dataset (obtained via (Road Sign Detection))

A comparison of a bar graph

Description automatically generated

Figure 2. Distribution of Object Classes in the Dataset

### Data Preprocessing

#### The data was split into a training and a validation set with the ratio of 4:1. As a result, the training set contained 701 images and the validation set had 176. Additionally, since the original format of the annotations were not ready to be used in the training pipeline, label format conversion was performed. Since the models used in the experiments required two different training frameworks, there were two types of conversion. The first framework was the TensorFlow object detection API which required the annotations in TFRecord format, while the Ultralytics framework for YOLOv5 expected the annotations in text format with each line describing the bounding boxes of the objects. More details of the frameworks are discussed in Section 2.2. It is notable that the training and validation sets had separate annotation conversion, so that resulting files were distinctly addressed in the training configuration to be the training and validation annotations respectively. Further details of the data preprocessing steps can be found in the Jupyter notebooks.

### Data Augmentation

Data augmentation is a technique that helps diversify the training data by adding variations to the images. For fair benchmarking and evaluation, similar augmentations had to be used in both frameworks (e.g., if the TensorFlow models used shearing at 30o, YOLOv5 had to use the same transformation), meaning that we could only choose the augmentation options that were available in both frameworks. Additionally, in the context of traffic sign detection, some transformations would not be applicable such as vertical or horizontal flips, or 90o rotation. Due to these requirements, there were a limited number of options available that could be utilized. In this project, we could select only one augmentation option that fitted the situation, and it was image scaling (Figure 3, Figure 4). Nevertheless, this single augmentation proved to be effective as described in detail in a later section of this report.

A screenshot of a computer program

Description automatically generated

Figure 3. Data Augmentation Configuration for the TensorFlow Framework

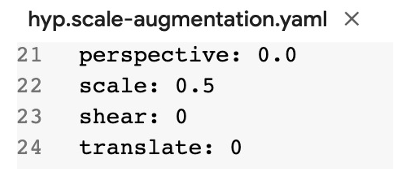


Figure 4. Data Augmentation Configuration for the YOLOv5 Framework