Detecting and Classifying Traffic Signs Using Deep Learning

By: Aanya Lakhani

2/26/2024

Abstract

In the world of technology and AI, several different companies have tried to bring the concept of self-driving cars to life. The purpose of this research was to train an AI model to efficiently detect and label different traffic signs (stop, traffic light, crosswalk and speed limit). A labeled dataset of 877 images with bounding boxes was used. The dataset was balanced by undersampling the overrepresented class (stop sign). This reduced the size of the dataset to 252 images. The images were cropped according to their bounding boxes to minimize distractions (only done for classification). The dataset was split into a training (80% of dataset) and validation (20% of dataset) list. For object detection, the images were not cropped. Five different computer vision models were used for classification. The Logistic Regression had the best accuracy with 93.2%, followed by the K-Nearest Neighbors Classifiers with an accuracy of 89.8%. The Decision Tree Model did well too, with an accuracy of 78.0%. The Convolutional Neural Networks, a deep learning model had an accuracy of 79.6% The Perceptron Model had the lowest accuracy of 32.8%. For object detection, YOLOv8 with the COCO Metrics BackBone was used. The data was first converted into the bounding box format (xyxy) then the model was trained. The model was tested on individual images. Object detection had a validation loss of 4.47, but was not able to detect traffic signs well.

1. Introduction

There have been many efforts to create autonomous cars that can accurately navigate through the road safely. One of the biggest problems faced by autonomous cars is making decisions in the presence of distractions or in poor lighting. In unpredictable conditions, it can be hard to detect and identify objects on the road. In this study, a classification and object detection model was created to be able to detect and classify traffic signs. A dataset by Larxel, was downloaded from Kaggle. The dataset consisted of images, their labels, and their bounding boxes. The bounding boxes were used to train the model for object detection. Only supervised learning was used where the model trained on both the images and their corresponding labels. The model classified images based on the labels, 'Stop Light', 'Speed Limit', 'Crosswalk', and 'Traffic Light'. For testing, the model was given images, and it had to classify them according to the labels. The models used for classification were Perceptron, Decision Tree, Convolutional Neural Networks (CNN), Logistic Regression, and K-Neighbor Classifier (KNN). For object detection, the YOLOv8 model was used with the COCO Metrics Callback backbone.

2. Background

Traffic sign detection in real time is not an easy task, especially because of the amount of distractions that exist. This task was attempted by Mr. Chang-il Kim et al. They experimented with a YOLOv3, YOLOv4, YOLOv5 detector and strongSORT tracking model.

Their main aim was real-time traffic sign detection in urban areas. They tried to capture several different environmental conditions that could affect traffic sign detection, like gloominess, rain, sunlight and cloud. After collecting data, they realized that their dataset was highly unbalanced. The smallest class had 61 images and the biggest class had 238,857.

The YOLOv4 detector performed the worst, while the YOLOv5 detector performed the best, with a mean average precision of 0.86, when the weather conditions were perfect. Rainy conditions performed better than perfect conditions and cloudy weather performed the worst out of all the other weather conditions.

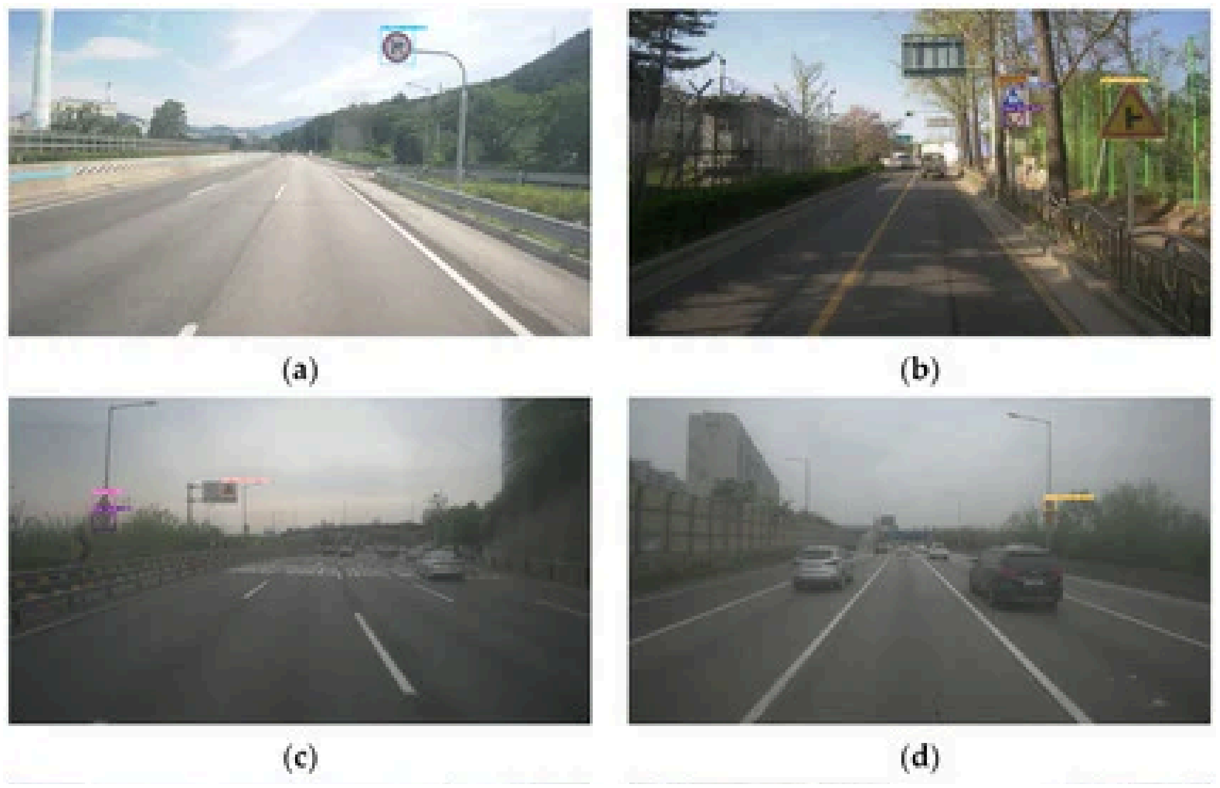


Figure 1: Shows the detections in rainy conditions.

Compared to a lot of other studies that have been done on real-time traffic sign detection, this one does significantly better with making the dataset as diverse as possible. Many other studies do not consider different weather conditions which would make it hard for detection.

3. Dataset

The dataset originally had 877 images. It consisted of the image, label, and coordinates for the bounding box around the traffic sign in the image. Many images contained more than one traffic sign.

There were many distractions in the dataset which would make classification very hard. This is why the images were cropped according to the bounding box. This made the focus of the image the traffic sign. This change was only made during classification and not during object detection. If an image contained more than one image, each traffic sign in the image was cropped and saved individually. This increased the size of the dataset. After preprocessing, the size of the dataset increased to 1,244 before balancing.

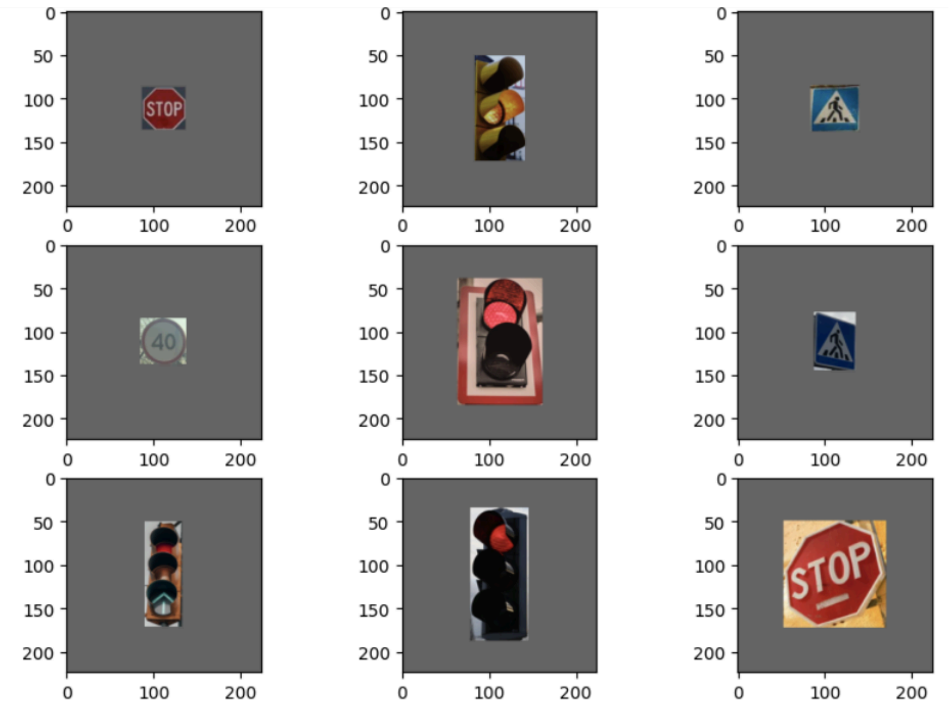


Figure 2: Shows the images after pre-processing.

One problem faced was that sometimes the traffic sign was very far and hence the image was very pixelated after preprocessing. To conquer this, all images that were smaller than 20x20 pixels were deleted. This reduced the size of the dataset to 688 images.

While training, because of the gray space, the model wasn't performing well. To tackle this, the images were stretched out and the size of them was decreased to 150x150 px.

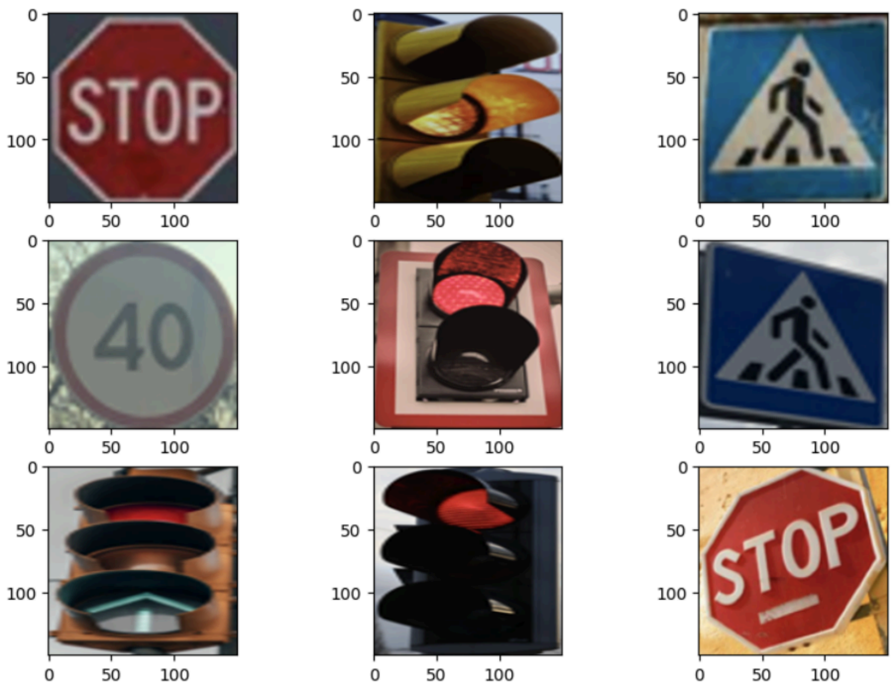


Figure 3: Shows the images after stretching them out to fit the size.

For object detection, no major changes were made to the dataset. The images were resized to 224 x 224 pixels, and the coordinates of the bounding boxes were changed accordingly.

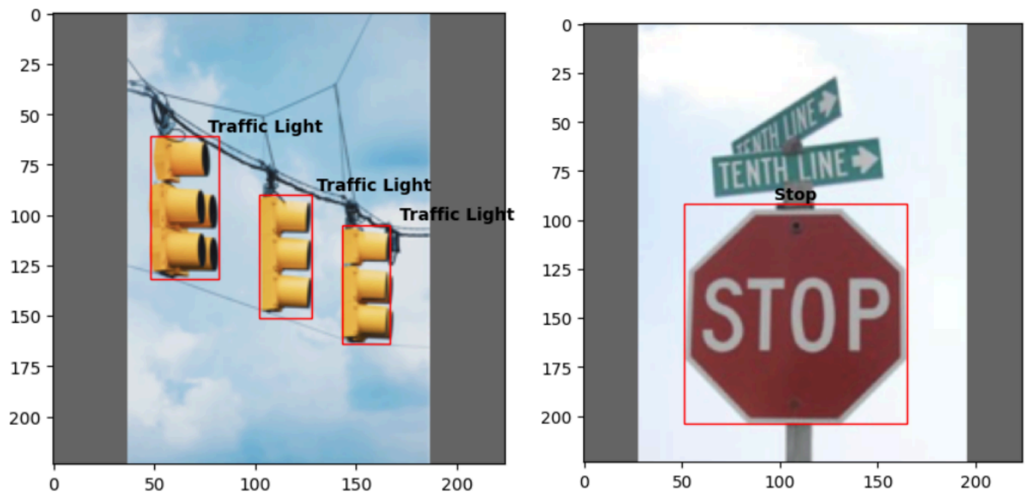


Figure 4: Visualizes the dataset, including the bouncing box and labels.

During classification, the dataset was highly unbalanced. There were 99 crosswalks, 74 stop signs, 443 speed limits, and 72 traffic lights combined, in all the 688 images. The overrepresented classes (speed limit, traffic light, and crosswalk), were cut in an attempt to make the number of all the classes the same. This reduced the dataset size to 296 images.

4. Methodology/Models

4.1 Classification

4.1.1 Getting the Data

The dataset for this project was downloaded from Kaggle, and uploaded into Google Drive. Google Drive was mounted onto Google Colab from where it was accessed.

4.1.2 Splitting the Dataset

The cropped images and the labels were split into a train and test, using scikit-learn's train_test_split(). 80% of the data was used for training and 20% of the data was used for testing. By splitting the data into a training and testing set, the model can be evaluated by how it works on new data.

4.1.3 Models Used

In total, five different classification models were used. The models experimented with for classification are: Decision Tree, Perceptron, Convolutional Neural Networks (CNN), K-Nearest Neighbors (KNN), and Logistic Regression.

4.1.4 How the Models Work

The Decision Tree algorithm is a regression model that splits the data into nodes. This process continues based on the best attribute the model is able to find (Geeks For Geeks). This process continues until the nodes can't be split anymore. The model chooses the sub-node with the most pure data, or most uniform data.

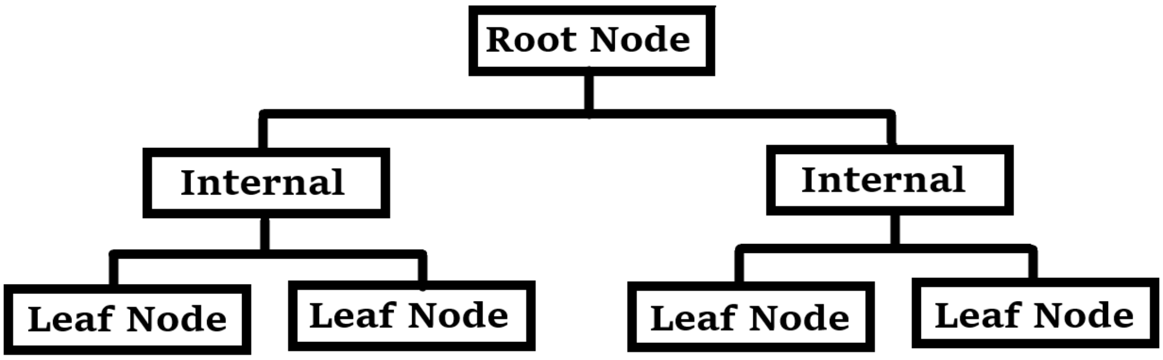


Figure 5: Shows how a decision tree model works.

The KNN model classifies images by trying to find the closest class to the image based on all the other data.

The Perceptron model is a neural network model that mimics the function of neurons in a brain. When it receives input, it multiplies the inputs and the weights together. It then substitutes that value into an activation function to get an output (Geeks for Geeks & Banoula).

The CNN algorithm is a deep learning model that works by identifying patterns in images.

4.2 Object Detection

4.2.1 Splitting the Dataset

For object detection, the same train_test_split() function was used. But instead of splitting the cropped images and the labels, the original images (with one or more traffic signs), labels and bounding boxes were split into a train and test set. The labels were converted into numbers (0: traffic sign, 1: crosswalk, 2: speed limit, 3: stop sign)

4.2.2 Model Used and How it Works

The YOLOv8 model with the COCO metrics callback was used for object detection. The YOLO model works by breaking up the image into grids. It then calculates the probability of an object being in an image and creates bounding boxes accordingly. After creating bounding boxes, a Non-Maximum Suppression technique is used to remove all the bounding boxes that are copies of each other or do not contain an actual image.

5. Results and Discussion

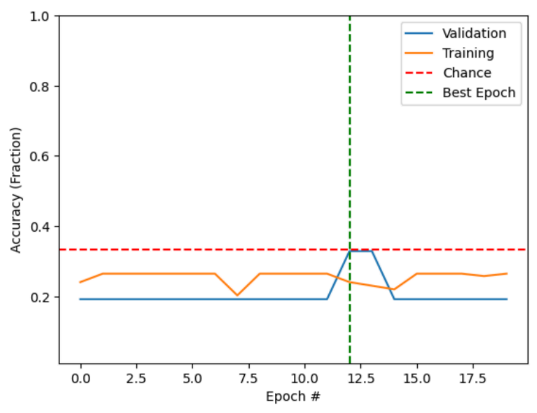
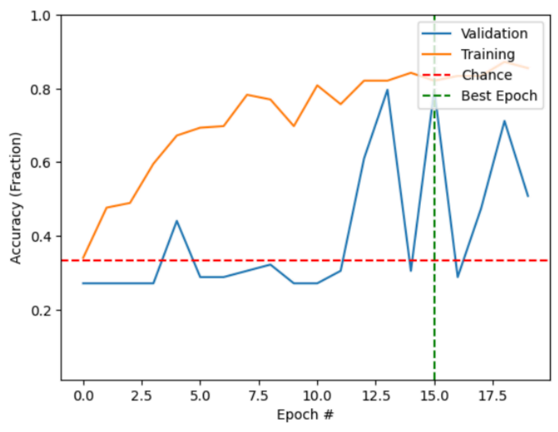
5.1 Classification

Logistic Regression did the best with an accuracy of 93.2%, followed by the KNN Classifier with an accuracy of 89.8%. The CNN model had an accuracy of 79.6%. The Decision Tree model was not far behind with an accuracy of 78.0%. The Perceptron model did the worse with an accuracy of 32.8%.

Decision Tree Test Labels:
[0., 0., 3., 1., 3., 3., 2., 0., 2., 0., 1., 1., 2., 1., 3., 2., 1., 2., 2., 0., 3., 3., 2., 2., 3., 1., 1., 0., 3., 1., 1., 0., 3., 3., 0., 3., 1., 2., 0., 0., 1., 1., 0., 1., 3., 2., 1., 2., 0., 0., 2., 0., 1., 1., 1., 3., 0., 0., 2.]

Decision Tree Output Labels:

[0., 0., 3., 2., 3., 3., 2., 0., 2., 0., 1., 1., 2., 1., 3., 2., 1., 1., 2., 1., 2., 3., 0., 3., 3., 2., 1., 2., 3., 1., 1., 0., 3., 3., 0., 3., 1., 2., 0., 0., 2., 1., 0., 1., 3., 2., 2., 2., 1., 3., 2., 1., 1., 1., 1., 3., 0., 0., 2.]

Perceptron	CNN
 **Figure 6:** Shows the progress of the Perceptron Model over 20 epochs. The highest accuracy was 32.88%.	 **Figure 7:** Shows the progress of the Perceptron Model over 20 epochs. The highest accuracy was 79.6%.

The Perceptron and CNN worked the best with a learning rate with a learning rate of 0.001. A higher learning rate caused the models to overfit even more, and the validation accuracy decreased too.

The KNN Classifier performed the best with 1 neighbor. It performed the worst with 14 neighbors with an accuracy of 66.1%.

5.1 Object Detection

Object Detection did not work well, not able to detect even if the traffic sign occupied the whole image. It got a validation loss of 4.48 and a validation box loss of 3.55.

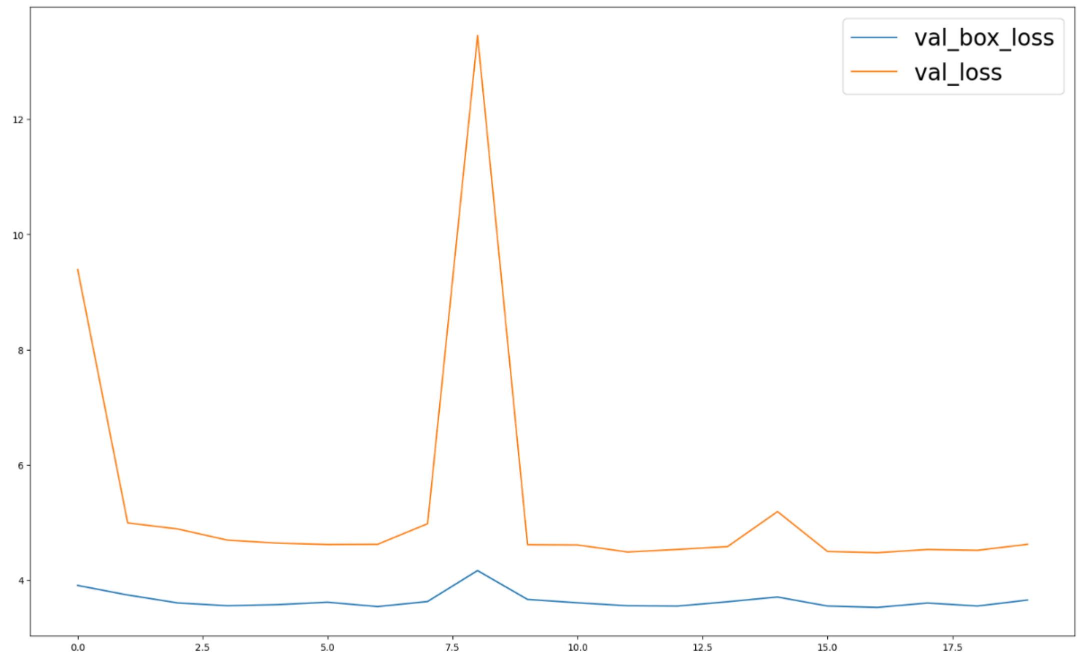


Figure 8: Shows the validation loss and validation box loss of the object detection model over 20 epochs

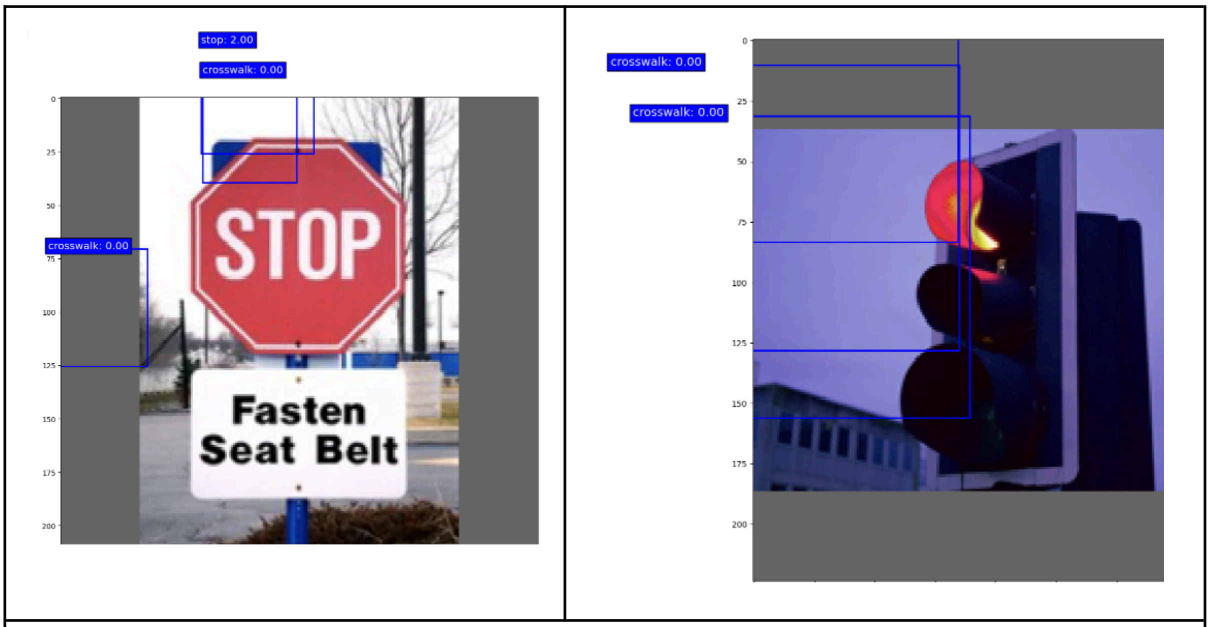


Figure 9: Shows an example of object detection.

6. Conclusion

This project went over organizing, pre-processing, and splitting the dataset. Additionally, it went through training five different classification models, and one object detection model.

For classification, although the model was not trained on images with a lot of distractions, without them it performed very well.

To prevent overfitting with classification, a dataset with more images would be ideal. Another way to increase the size of the dataset would be to take another dataset and merge it together with the existing one. This would be beneficial for object detection too.

Although the aim of accurate object detection wasn’t fulfilled, a bigger dataset and more training would greatly help to accomplish this objective.

Furthermore, using this same dataset, the next task would be to try to use more advanced classification models that can classify images in the presence of distractions. Additionally, working on the object detection model would make this model extremely useful in the real world.

Acknowledgements

I would like to thank my mentor, Ms. Kahye Song, who helped me find a dataset, debug and also guided me on the path that I should go on. Also, I would like to thank my parents who provided me this opportunity to work with Inspiriit AI.

References

Banoula, M. (2023, May 10). *What is Perceptron: A Beginners Guide for Perceptron*. Simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=The%20Perceptron%20receives%20multiple%20input,the%20class%20of%20a%20sample.

GfG. (2023, December 4). Decision tree in machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree-introduction-example/

GfG. (2023a, October 13). *Perceptron class in Sklearn*. GeeksforGeeks.

https://www.geeksforgeeks.org/sklearn-perceptron/

Kim, C., Park, J., Park, J. S., Jung, W., & Lim, Y. (2023). Deep Learning-Based Real-Time Traffic Sign Recognition System for urban environments. Infrastructures, 8(2), 20. https://doi.org/10.3390/infrastructures8020020

Dataset: *Road sign detection*. (2020, May 24). Kaggle. https://www.kaggle.com/datasets/andrewmvd/road-sign-detection