Traffic Light Detection for Autonomous Vehicles

Reshma Nawaz, Kaisen Ye

Viterbi School of Engineering

University of Southern California

December 7th, 2023

ABSTRACT

In this project, we dive into the development and comparison of two advanced object detection systems, Faster R-CNN and YOLOv3, aimed at enhancing traffic light detection in autonomous and semi-autonomous vehicles. Our study focuses on leveraging Convolutional Neural Networks (CNNs) for accurately detecting and classifying traffic signals in varied urban driving scenarios, crucial for safe navigation and adherence to traffic laws.

We detail the design and implementation of these models, highlighting the challenges encountered and solutions adopted during the process. The Faster R-CNN model, utilizing a MobileNetV2 backbone, shows promise in learning and adapting to different traffic conditions, although it faces challenges in achieving optimal precision and recall during validation phase. The YOLOv3 model, with its distinct architecture, also presents potential in object detection despite the challenges faced in training due to computational constraints.

This project not only contributes to the field of autonomous vehicle technology but also sets a foundation for future advancements in traffic light detection systems, proposing potential areas for further research and development.

1. INTRODUCTION

In the swiftly evolving landscape of autonomous and semi-autonomous vehicles, traffic light detection remains a cornerstone for ensuring safe and efficient navigation. The ability of self-driving cars to accurately detect traffic signals is fundamental for adhering to traffic laws and preventing road incidents. This project aims to address this pivotal aspect of autonomous technology. Our focus is on developing a robust model capable of recognizing traffic lights under a variety of conditions, thereby enhancing the safety and dependability of autonomous driving systems and Advanced Driver-Assistance Systems (ADAS).

The integration of sophisticated traffic light detection systems into vehicles is not just a technological advancement but also a significant stride towards better urban traffic management. By facilitating efficient navigation, these systems can significantly reduce congestion, improve traffic flow in smart cities, and benefit logistics and transportation sectors. The broad spectrum of this technology's impact encompasses enhancing route planning for companies like FedEx and UPS, thereby reducing operational delays and increasing overall efficiency. Our project's outcomes have the potential to refine the operation of self-driving vehicles across diverse environments, from bright daylight to challenging conditions like storms or snow, ensuring both driver and pedestrian safety.

Our approach leverages Convolutional Neural Networks (CNNs) and advanced object detection algorithms, such as Region-based Convolutional Neural Networks (R-CNNs) and You Only Look Once (YOLO). These techniques are pivotal for extracting image features and locating traffic lights in various visual scenarios. CNNs' capability in image recognition, combined with the precision and real-time processing of R-CNNs and YOLO, forms the basis of our model. This project draws inspiration from seminal works such as Fast R-CNN by [1] and improved YOLOv4 algorithm by [2], aiming to build upon their foundational research to create a more versatile and accurate traffic light detection system.

2. PROBLEM STATEMENT

In the pursuit of enhancing autonomous vehicle navigation, our project aims to develop a machine learning system capable of detecting traffic lights and accurately determining their position and color in a frame. The system will analyze static images captured from a stereo camera mounted on a vehicle. These images, with a resolution of 1280 x 960 pixels, depict various urban driving scenarios in Pacific Beach and La Jolla, San Diego, under different lighting and weather conditions, including both day and night sequences.

A critical aspect of this project is to ensure the system's robustness across these varying environmental conditions, focusing particularly on the challenges posed by changing light and weather. The dataset utilized for this purpose is approximately 5 GB in size, encompassing a diverse range of traffic scenarios.

Given the large size of the dataset and the complexity of the task, the system will heavily rely on GPU capabilities to process the data efficiently. While the primary objective is to work on static frames, the system must be scalable and efficient enough to potentially adapt to real-time processing in future iterations.

The success of this project hinges on its ability to not only detect traffic lights within these diverse frames but also accurately distinguish between different light colors and pinpoint their precise location. This capability is crucial for the safe and efficient operation of autonomous vehicles in a variety of driving conditions.

3. MINI LITERATURE REVIEW

[1] introduces Mask R-CNN, an extension of the Faster R-CNN framework, adding a branch for object mask prediction parallel to the bounding box recognition. Mask R-CNN has proven to be highly effective in precise object detection and segmentation, making it particularly suited for traffic light detection where accuracy in identifying and delineating the traffic light within a frame is paramount.

[2] offers an extensive overview of deep learning applications in sensor-based activity recognition, highlighting the advancements in algorithms like YOLO and their adaptations for various real-world applications. It underscores the potential of deep learning in extracting meaningful patterns from complex sensor data, relevant to our project's focus on detecting traffic lights from camera images under varying conditions.

[3] presents the Faster R-CNN framework, a groundbreaking advancement in object detection. Faster R-CNN innovatively integrates a Region Proposal Network (RPN) with the Fast R-CNN model, creating a unified system for efficient and accurate object detection. This architecture addresses the bottleneck in region proposal computation, substantially accelerating the detection process without compromising accuracy. The paper demonstrates the effectiveness of Faster R-CNN across various challenging datasets like PASCAL VOC and MS COCO, showcasing significant improvements in detection speed and accuracy. This work has been pivotal in the field, setting a new standard for real-time object detection systems.

4. DATA PREPROCESSING

The LISA Traffic Light Dataset is a 5 GB dataset structured into annotated traffic light data captured over 44 minutes of video sequences, consisting of 43,007 frames with 113,888 annotated traffic lights for day and night training images, with corresponding annotations. We meticulously traversed these directories to collate paths for annotation files and image directories. Annotations were aggregated into a DataFrame, consolidating over 40,000 images for streamlined processing.

4.1. Resizing

Initially, our preprocessing pipeline incorporated a transformation to resize the images to a uniform dimension of 800x800 pixels. This step was intended to maintain consistency in the input data, which is typically beneficial for object detection tasks. However, this dimension standardization led to suboptimal evaluation metrics during validation. Upon further investigation, we identified the crux of the issue: the bounding box coordinates and labels had not been appropriately scaled to match the transformed image dimensions.

This oversight meant that the spatial correspondence between the original annotations and the transformed images was lost, leading to inaccurate training and, consequently, poor model performance. Recognizing this, we reverted to using the original image resolution of 1280x960 pixels. This decision allowed us to maintain the integrity of the bounding box coordinates and labels, thereby aligning the training data more closely with the real-world conditions the model would encounter. This adjustment was crucial in improving the validation metrics and enhancing the overall efficacy of the Faster R-CNN model for traffic light detection.

4.2. Labeling

To achieve our goal of accurately detecting the position and status of traffic lights, we transformed the target annotations into a structured dictionary format. This reformatting included two key components: 'boxes', representing the coordinates of the bounding boxes, and 'labels', denoting the distinct statuses of the traffic lights such as 'stop' or 'go'. Each label was assigned a 7 distinct numerical value {'go', 'warning', 'stop', 'goLeft', 'warningLeft', 'stopLeft'}, facilitating efficient processing and classification by our model. This methodical approach ensured a precise and systematic representation of the target data, crucial for the effective training of our detection model.

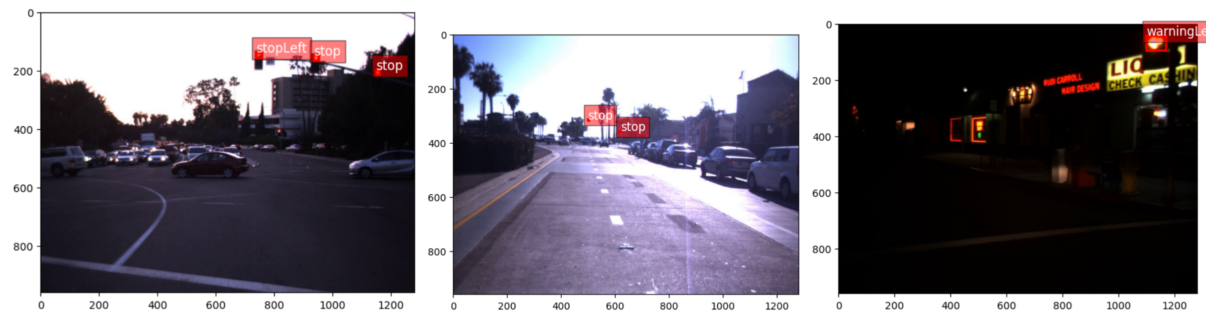

Figure 1: Sample images with labels

4.3. Data Handling

The dataset was divided based on the source of the data – 'dayTrain' and 'nightTrain'. Each of these subsets was then split into training and validation sets with an 80% and 20% ratio, respectively. This division was strategically implemented to ensure a comprehensive learning process, accommodating variations in lighting and environmental conditions.

For the purpose of testing, we utilized separate datasets, specifically 'sample-dayClip6' and 'sample-nightClip1'. This segregation of the testing data from the training and validation sets ensures an unbiased assessment of the model's performance in real-world scenarios, encompassing both day and night conditions.

Dataset	**Sample Size**
Training Dataset	14,510

Validation Dataset	3,628
Testing Dataset	910

4.4. Data Loader

The conventional PyTorch DataLoader, which aggregates all target dictionaries within a batch, presented a significant challenge by inducing errors in the training loop, incompatible with the Faster R-CNN framework. To address this issue, we developed a bespoke collate function. This function meticulously processes each target dictionary individually, thereby facilitating accurate batching and ensuring seamless compatibility with our chosen model. This tailored solution was pivotal in maintaining the integrity and efficiency of our training process.

4.5. Choice of Model

Given the lack of 'Mask' data in the LISA dataset and the considerable challenge of manually annotating a vast number of images, our team opted to employ the Faster R-CNN model [3] instead of Mask R-CNN [1]. This strategic decision was driven by a desire to align our approach more effectively with the available dataset and the practical constraints of our project. Utilizing Faster R-CNN allowed us to circumvent the limitations posed by the dataset and focus on achieving our primary objectives within a feasible framework. Additionally, our team researched and built out the architecture of the YOLO-v3 mode. YOLO (You Only Look Once) is a popular object detection algorithm that's known for its speed and accuracy. YOLO-v3 is a single neural network for detection, it is faster and more efficient than traditional models. The team had originally planned to implement YOLOv4 which is better suited over YOLOv3 in various aspects, however, due to its complexity in implementation and limited support from libraries the team decided to implement the YOLOv3 model due to its advantages. YOLOv3 is simpler and more straightforward in terms of implementation compared to YOLOv4. Its architecture and training process are easier to understand and work with, making it easier for beginners. Additionally, whilst the implementation was on Kaggle which provides a limited GPU space the YOLOv3 was preferred in terms of resource efficiency. Lastly, since YOLOv3 has been around longer than YOLOv4 it is more widely adopted and integrated into various frameworks and applications.

5. EXPERIMENTS AND RESULTS

In pursuit of an enhanced methodology for traffic light detection, our study embarked on a comparative analysis of two distinct yet state-of-the-art object detection architectures: Faster R-CNN and YOLOv3. We firstly explain the Faster R-CNN one.

5.1. Faster R-CNN

Faster R-CNN is renowned for its integration of a Region Proposal Network (RPN) with the detection network, streamlining the process to enhance both speed and accuracy. The architecture commences with a backbone of convolutional layers, which in the context of our implementation, is instantiated by MobileNetV2, selected for its lightweight yet powerful feature extraction capabilities.

In our research, we encapsulated the MobileNetV2 within a custom class, 'MobileNetV2Backbone', which loads the pre-trained network and constructs the foundational feature map from the input image. The strength of MobileNetV2 lies in its ability to capture a wealth of hierarchical features critical for identifying objects across varied scales and aspect ratios.

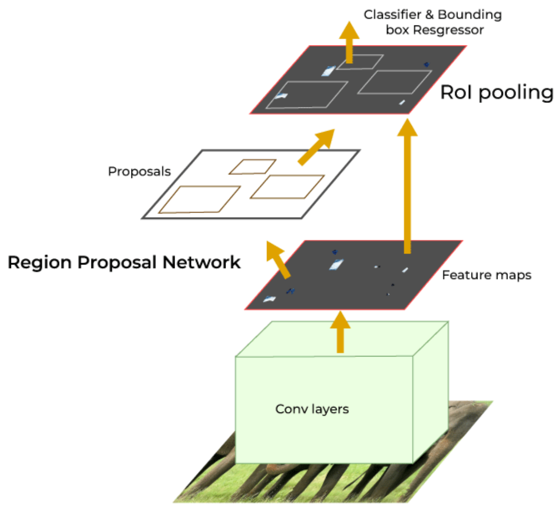

Figure 2: Faster R-CNN architecture

Our implementation of Faster R-CNN employs an AnchorGenerator configured with multiple sizes and aspect ratios, catering to the diverse shapes and orientations of traffic lights. We trained the model over 20 epochs, applying a learning rate of 0.005 and employing the Adam optimizer for its robustness in handling noisy gradients. A learning rate scheduler fine-tunes this rate, adapting it to the model's learning pace.

The training loop processed images and targets in batches, updating the model weights based on the computed losses. This batch processing allowed for efficient memory usage and faster computations, vital for handling large datasets.

The model's performance is gauged through custom evaluation functions that calculate Intersection over Union (IoU) with a threshold of 0.5 and evaluate precision and recall metrics by matching predictions against ground truths. These functions are critical for tuning the model's detection sensitivity and ensuring that the RPN and the detection network synergistically refine the prediction accuracy.

The training loss demonstrated a consistent downward trend across the epochs, a clear indication that the model was effectively learning and adapting to the training data. However, the precision and recall metrics, while improving over time, remained relatively low. This suggested that while the model was becoming better at identifying traffic lights, its accuracy in these identifications was not as high as we aimed for.

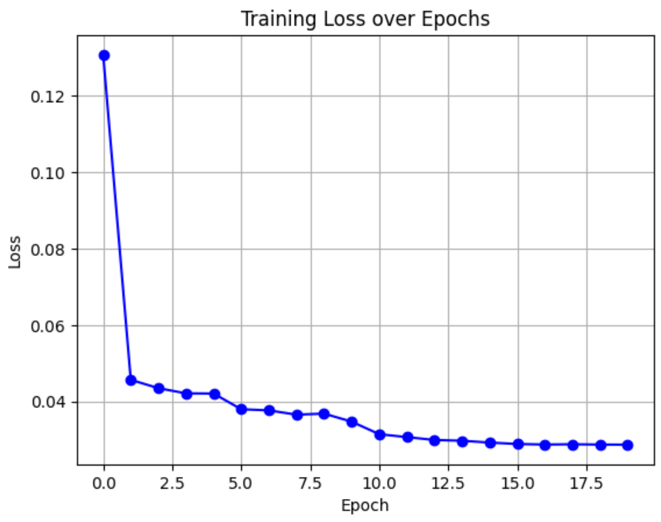

Figure 3: Faster R-CNN training losses

Similarly, the gradual increase in accuracy, although promising, was not at the level we had anticipated. This pointed towards potential limitations in the model's architecture or the need for more nuanced tuning to fully optimize its performance.

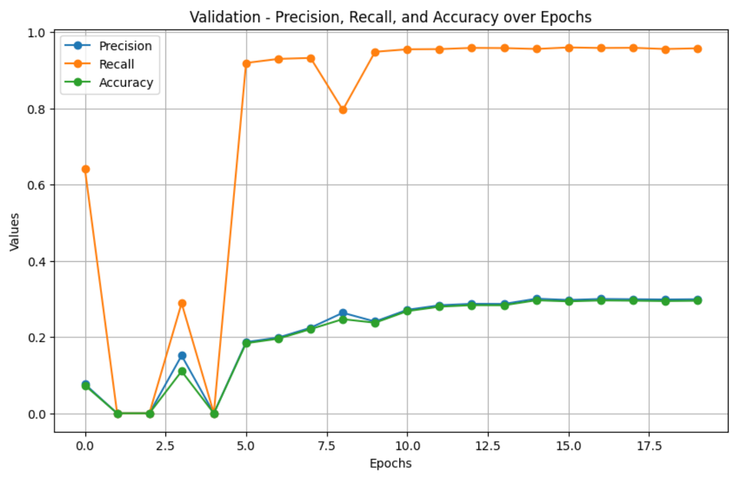

Figure 4: Faster R-CNN precision, recall, accuracy

5.2. YOLO-V3

The YOLO-v3 model employs a Convolutional Neural Network (CNN) architecture for object detection in images, structured into three main sections: the backbone, the neck, and the head. The neck merges features from different scales to enhance object detection accuracy. The head includes fully connected layers responsible for predicting both the location and class of each detected object in the image. The backbone consists of convolutional layers designed to extract image features. YOLOv3 utilizes anchor boxes, which are predetermined bounding boxes with varying sizes and aspect ratios. These anchor boxes are used in the prediction of the location and dimensions of objects within the image.

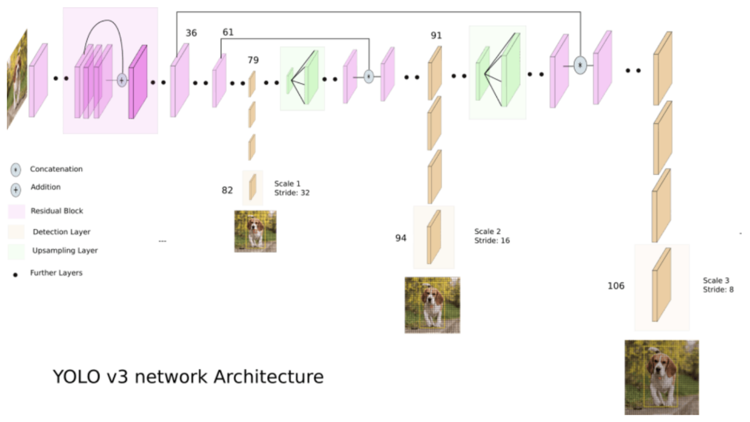

Figure 5: YOLO-v3 Network Architecture

Traditionally, the YOLO-v3 model uses the Darknet-53 as a feature extraction model. Darknet-53 consists of 53 convolutional layers. These layers are organized in a sequential manner, forming the backbone of the YOLOv3 network. Each layer performs convolutions, applying filters to the input data to extract hierarchical features. These features gradually become more abstract and complex as they progress through the network. Darknet-53 is a custom neural network architecture that is not part of the torchvision library or any other widely used Python libraries for deep learning due to which our team implemented the ResNet-50 for a simpler and beginner friendly implementation.

ResNet-50 is a Convolutional Neural Network (CNN) architecture that is able to be the backbone of the detection model YOLO-v3. The ResNet-50 is a variant of the Residual Network architecture which includes 50 layers. As the backbone of YOLOv3, ResNet-50 is responsible for extracting features from input images. In implementations of this model, ResNet-50 is chosen as an alternative choice in comparison to Darknet-53 due to its effectiveness in feature extraction plus its availability in libraries like PyTorch and TensorFlow.

The data was loaded in the similar manner as for the Faster RCNN Model where it is split into 7 classes stop, stopLeft, go, goLeft, warningLeft, warning, background. The annotation boxes are divided into bounding box coordinates bounding boxes as four values: x_center, y_center, width, and height. This step is necessary to transform data from the standard format to the YOLO format.

- X_center: x-coordinate of the center of the bounding box which is relative to the width of the image representing the horizontal position of the center of the object.
- Y_center: y-coordinate of the center of the bounding box which is relative to the height of the image representing the vertical position of the center of the object.
- Height: height of the bounding box which is relative to the height of the entire image which is useful in identifying how tall the object is in relation to the height of the image.
- Width: width of the bounding box comparison to the image height.
- Corresponding formulas:
  - x_center = (xmin + xmax) / 2 * image_width
  - y_center = (ymin + ymax) / 2 * image_height
  - width = (xmax - xmin) * image_width
  - height = (ymax - ymin) * image_height

The implementation of the model includes the loss function which was given as the softmax function. It computes the cross-entropy loss between the predicted class probabilities and the target labels. Additionally, the Adam optimizer was used to handle the noisy gradients along with a L2 regularization to reduce overfitting. The entire model was trained over 10 epochs with a learning rate of 0.0001. Due to the format of the dataset along with the complexity of the model, our team was unable to successfully run the model to train. Our team encountered a batch sizing issue that persisted despite attempts to resolve it. This challenge arose as we referenced the architecture of YOLO and experimented with various backbone structures, iteration sizes, learning rates, etc. Due to the GPU contractions on Kaggle, the team was unable to consistently run the model to test for improvement.

The team encountered a batch sizing problem during model training that persisted despite numerous attempts to resolve it. We tried various approaches, including adjusting image sizes, experimenting with different backbone architectures, modifying the batch size, and consulting various research papers and communities. Unfortunately, despite these efforts, we were unable to overcome this issue, resulting in the inability to train our model effectively.

6. CONCLUSION AND FUTURE WORK

In concluding our investigation into traffic light detection, the implemented Faster R-CNN model, leveraging the MobileNetV2 backbone, represents a significant step towards the integration of advanced object detection systems in autonomous vehicles.

Throughout our study, we have observed the model's aptitude for learning and adapting to diverse traffic scenarios. Yet, the journey towards an impeccable detection system is far from complete. The model we constructed, despite its architectural strengths, displayed limitations in achieving the high levels of precision and recall necessary for real-world application.

6.1. Future Work

Potential future studies might explore the integration of more sophisticated backbones like EfficientNet or Vision Transformers, which we were not able to experiment due to time constraints, could potentially capture richer representations of the input space, leading to better detection performance. Furthermore, future iterations of the system could explore the fusion of data from multiple sensors, such as LiDAR and radar, to complement the visual cues and to potentially improve detection accuracy. In terms of the YOLO model, in future implementations we will try to further reduce the batch size, make modifications to the model architecture by implementing a different backbone and alter the number of layers being processed. Additionally, we will use a different GPU with a larger memory capacity.

7. REFERENCES

[1] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, DOI: 10.1109/ICCV.2017.322.

[2] Z. Wang, C. Yu, W. Zhao, Y. Zhu, M. Liu and Y. Zhao, "Deep Learning for Sensor-Based Activity Recognition: A Survey," in Sensors, vol. 22, no. 1, 200. DOI: 10.3390/s22010200.

[3] Ren, S., He, K., Girshick, R., & Sun, J. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", 2015 arXiv:1506.01497. DOI:10.48550/arXiv.1506.01497.