Traffic Light Detection Using Deep Learning Methods

https://github.com/seanzero7/TrafficLights

Sean Hall, Henry Miller, Reid Miller, Simon Yung

Abstract:

This study presents a dual-convolutional neural network system for traffic light detection, focusing on applications in autonomous vehicles. The first network utilizes the YOLO (You Only Look Once) model to locate traffic lights within images captured by vehicle-mounted sensors. In contrast, the second network classifies the detected lights into red, yellow, or green categories. The YOLO model achieves 60% recall, 80% precision, 69% mAP50, and 39% mAP50-95, while the classification model attains over 99% validation accuracy. This research highlights the potential of combining advanced object detection and classification techniques to enhance autonomous vehicle navigation, emphasizing the importance of robust datasets, diverse environmental conditions, and computational resources.

Introduction:

Traffic light detection is an essential component of partially and fully autonomous vehicles. To guarantee driver and pedestrian safety, systems must be implemented to ensure that autonomous vehicles can accurately and efficiently detect traffic lights' location and the type of light being identified. These systems also must be able to overcome challenges stemming from the variety of lighting conditions, weather conditions, and obstructions that vehicles will naturally encounter while driving.

In order to tailor our model to the two-pronged goals of traffic light detection, we have decided to implement two different convolutional neural networks. The goal of the first neural network is to determine the location of any potential traffic lights, while the goal of the second neural network is to identify the type of traffic light.

The input used for the first model will be a simple JPG image taken from a sensor placed at the front of a car. The first convolutional neural network will utilize the preexisting object detection model, YOLO (You Only Look Once). The model will identify the traffic lights' location in the image, returning the minimum and maximum x and y coordinates to form a bounding box around each light. These bounding box coordinates will then be used to crop the initial image, producing new images of each individual light. Each new image file will then be passed into a second convolutional neural network outlined below, which will determine the type of light used.

The results show a relatively high level of success. The YOLO light detection model had 60% recall, 80% precision, 69% mAP50, and 39% mAP50-95. The light classification model showed an extremely high level of accuracy with a validation accuracy of over 99%.

Background:

Our deep learning project was inspired by the transformative potential of self-driving vehicles and the profound role artificial intelligence (AI) and deep neural networks play in their development. The

rapid advancements in AI-driven technologies signal a future where self-driving vehicles are not just a possibility but a cornerstone of transportation innovation.

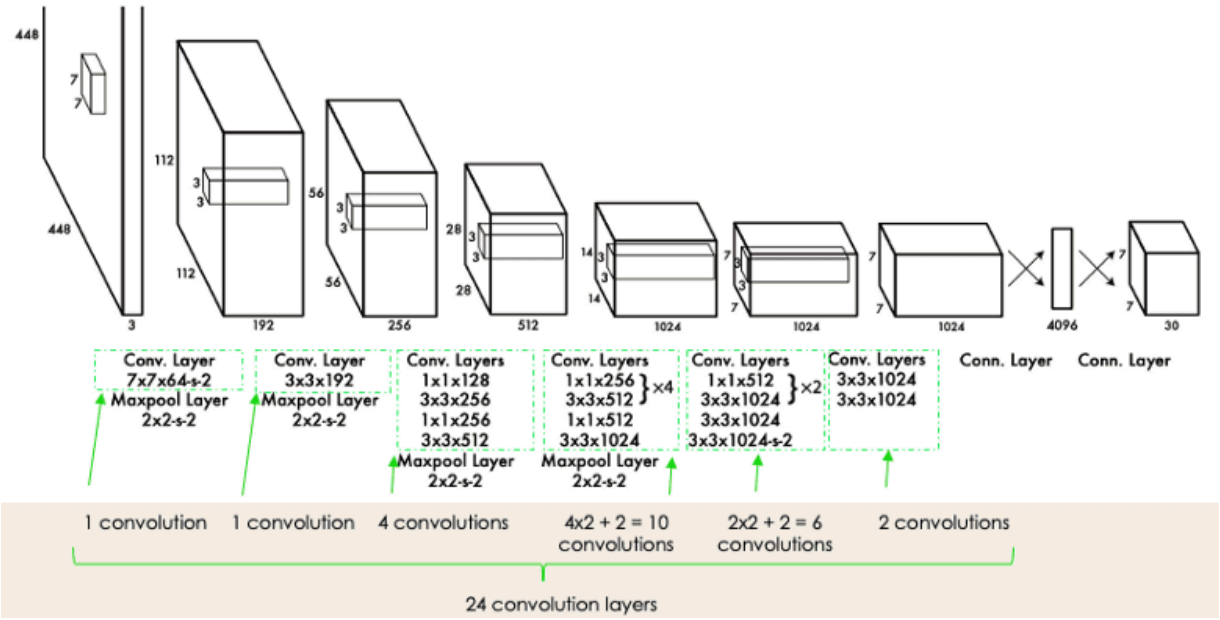
One of the most compelling advocates for this vision is Elon Musk, who has championed eliminating radar technology in favor of pure image processing for Tesla's Full Self-Driving (FSD) system despite the initial safety challenges it presented (Dnistran 2023). His bold stance reflects the growing confidence in computer vision and AI capabilities, which continue to evolve and achieve remarkable feats in accuracy and reliability.

Over time, the field of image processing has made significant strides, with breakthroughs in both hardware and algorithms making it more robust and efficient. Motivated by these developments and the broader vision of autonomous driving, our project seeks to contribute to this exciting domain. We designed and implemented a system that leverages AI for image segmentation and employs deep neural networks for classification, embodying the same principles that drive cutting-edge research and development in the industry. This work aligns with the overarching goal of advancing self-driving technology and exploring innovative approaches to AI-powered transportation solutions.

Methods:

The first traffic light detection and classification model created for this project was a single convolutional neural network with two outputs: a bounding box around each detected traffic light and a classification for the type of light. However, this method traded accuracy for simplicity, only correctly identifying light location and class around 50% of the time. As a result, we shifted towards a two-pronged approach to traffic light detection, allowing us to fine-tune each model to its specific task.

The project utilizes the YOLO (You Only Look Once) deep learning algorithm, a state-of-the-art model for object detection tasks. YOLO is implemented through the ultralytics library, which provides various versions of the algorithm optimized for high-speed and accurate detection. The algorithm operates by dividing the input image into a grid and predicting bounding boxes for objects within each grid cell, along with confidence scores representing the detected object's probability. This grid-based probabilistic approach enables YOLO to perform highly efficient end-to-end object detection.



To identify and localize objects, YOLO utilizes residual blocks, bounding box regression, intersection over unions, and non-maximum suppression. YOLO was specifically chosen for its high speed, detection accuracy, and past successes in traffic light detection (Sarhan and Al-Omary, 2022).

This project's traffic light detection dataset is sourced using the Kaggle API. Preprocessing steps include organizing images and annotations into training and validation sets and standardizing input formats using libraries such as CV2 and numpy. Additionally, we found that adjusting the size of the given bounding boxes improved the model's accuracy by scaling them up. The pipeline prepares the data to be fed into the YOLO model, ensuring compatibility and consistency for practical training.

Our color detection model is a Convolutional Neural Network (CNN) designed for classification tasks with three output classes: Red, Yellow, and Green. It consists of three convolutional layers (Conv2D) with increasing filter counts (32, 64, and 128) followed by max-pooling layers to reduce spatial dimensions. After the convolutional layers, the data is flattened into a 1D vector and passed through two dense layers, one with 128 units and another with three units (corresponding to the output classes). A dropout layer is included between the dense layers to prevent overfitting.

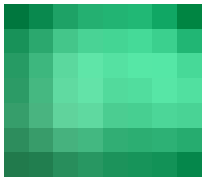
```
# Model 2 Architecture
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
```

```
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(label_to_idx), activation='softmax')
])
```

Experiment:

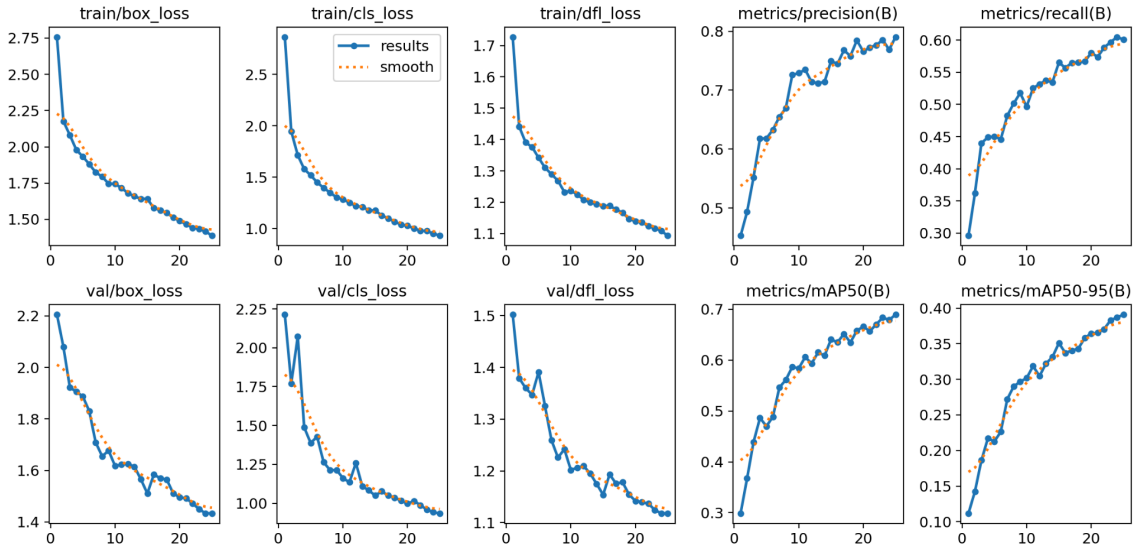
The dataset used in this machine learning project consists of images of traffic intersections, each associated with JSON files containing annotations for traffic light color detection. There were 3000 images, all with a multitude of traffic signals in each of them. The JSON data is structured to provide detailed information about the bounding boxes for each annotated object, which represent the regions containing traffic lights in the images. Each annotation includes the filename of the corresponding image, the coordinates of the bounding box (xmin, ymin, xmax, ymax), and additional properties such as the occlusion, truncation, and whether the object should be ignored in training.

We already ran into issues with the dataset, starting with the bounding boxes. The bounding boxes only encapsulate the light that was on (see original data on the right) instead of the whole traffic light. So, we had to scale every image to encapsulate the entire traffic light for proper identification. We scaled the x-values of the bounding box by 1.5 and the y-values by 2 so the whole traffic light would be encapsulated in the bounding box.



Traffic lights are further categorized by color and shape within the bounding boxes. The "color" attribute identifies the color of the light (e.g., red, green), while the "shape" attribute provides a unique identifier for the shape of the traffic light. The "bndbox" inside each "inbox" entry further refines the location of the traffic light within the image. Additionally, the dataset includes flags indicating whether a particular instance is occluded, truncated, or difficult to identify, which helps in training models to handle imperfect data. The "ignore" flag is set to 1 for particular entries, signaling that those specific annotations should be excluded from the training process. The value field, when present, can provide further contextual information, such as a difficulty rating or a placeholder for more specific categorization. (Because of the scaling and success of the model, eventually, the model was good enough that we could ignore the ignore flag).

We then subsequently trained the YOLO model. We used the pre-trained YOLO model and then input our data to train the model. The main results of the model are below.



The metrics to be optimized for this model are recall, precision, mAP50, and mAP50-95. The results are discussed further in the following discussion section.

Now, onto the second model. We repurposed a CNN model from the lecture. We changed the input to the images defined in the bounding box with their associated colors: red, yellow, and green. We also changed the layers a bit to make more sense with color identification. Because the model it was based on was so robust, little tweaking was needed to have over 99% accuracy.

Discussion:

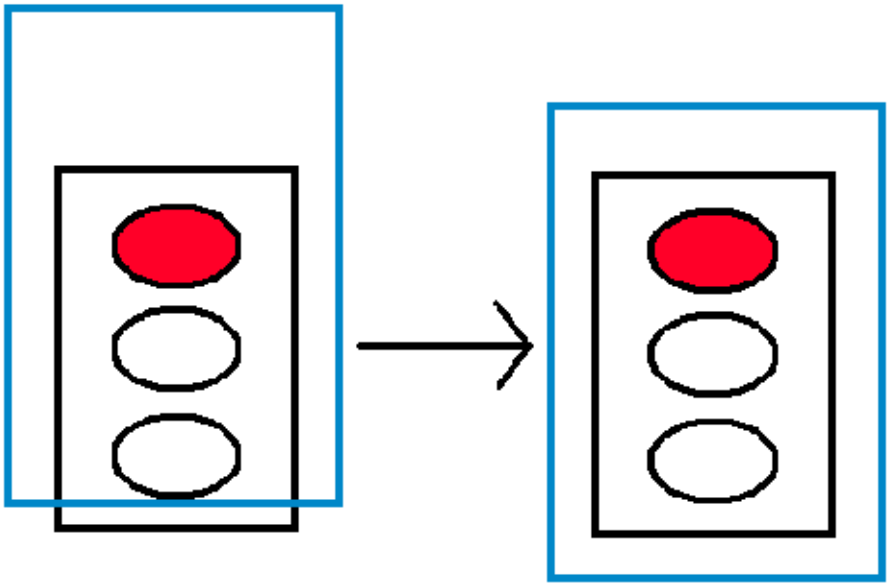
While the model was largely successful at predicting light location and classification, it is more powerful than the results suggest. This is due to the nature of the dataset used for training. Just over 2000 images were included in the dataset, with bounding boxes labeled specifically around the individual glowing bulb in each light. However, since surrounding information, such as the light's frame, is relevant, we rescaled the bounding boxes as described earlier.

Since a standard scaling cannot be applied to these bounding boxes in a way that perfectly fits every light, these new bounding boxes often contain extra information outside of the traffic light frame. Since this training bounding box information is slightly inaccurate in this way, a perfectly predicted output would register as slightly inaccurate as well. The graphic below helps illustrate this phenomenon. While the model would be ideal, the differences between these two boxes would still dampen result metrics.

The metrics to be optimized for this model are recall, precision, mAP50, and mAP50-95. The following block quote from the documentation explains the previously mentioned metrics:

- P (Precision): The accuracy of the detected objects, indicating how many detections were correct.
- R (Recall): The ability of the model to identify all instances of objects in the images.
- mAP50: Mean average precision calculated at an intersection over union (IoU) threshold of 0.50. It's a measure of the model's accuracy considering only the "easy" detections.
- mAP50-95: The average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95. It gives a comprehensive view of the model's performance across different levels of detection difficulty.

The observed mAP50 metrics appear less robust due to the scaling issues with the bounding boxes, as visualized below. However, the model's output bounding boxes outperform the input training bounding boxes because the IoU reference lacks precision. Given the inherent limitations of the provided data and input quality, achieving accurate metrics is challenging.



Input Training Bounding Box Output Bounding Box

Despite these limitations, a manual review of approximately 50 images revealed results that exceeded expectations. Even with lower mAP50-95 values, nearly all output bounding boxes demonstrated improvements over the input ones, as reflected in the accompanying visualization.

Consequently, while the numerical metrics may seem insignificant, our identification model performed exceptionally well in practice.

In addition to properly bounded training data, other dataset improvements could significantly impact accuracy. A larger dataset in even more diverse conditions would be beneficial. Integrating LiDAR and RADAR data could also increase accuracy. Including prior maps could also be helpful since the model could then specifically look for known locations.

Lastly, increased computational resources would be beneficial. Result metrics continually improved with more epochs, never reaching their inflection points. This suggests that further training, made possible by more computational resources, would continue to improve outcomes.

Conclusion:

This project successfully developed a traffic light detection and classification system using deep learning techniques, achieving promising results despite dataset limitations. The YOLO-based detection model effectively identifies traffic light locations, and the classification model demonstrates high accuracy. However, further improvements could be realized with expanded datasets, integration of additional sensor modalities like LiDAR or RADAR, and enhanced computational resources for extended training. This research underscores the transformative role of AI in advancing autonomous vehicle technologies, paving the way for safer and more reliable traffic systems.

References

Buqi, W. “Traffic Light Detection Dataset.” Kaggle, 19 Dec. 2021, www.kaggle.com/datasets/wjybuqi/traffic-light-detection-dataset.

Elon Musk Overruled Tesla Engineers Who Said Removing Radar Would Be Problematic: Report. InsideEVs, insideevs.com/news/658439/elon-musk-overruled-tesla-autopilot-engineers-radar-removal/. Accessed 10 Dec. 2024.

Redmon, Joseph, and Ali Farhadi. "YOLOv3: An Incremental Improvement." arXiv, 2018, https://doi.org/10.48550/arxiv.1804.02767.

Sarhan, N. H., and A. Y. Al-Omary. "Traffic Light Detection Using OpenCV and YOLO." 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 2022, pp. 604-608. IEEE, https://doi.org/10.1109/3ICT56508.2022.9990723.

Ultralytics. "YOLO Performance Metrics." Ultralytics YOLO Docs, 1 Oct. 2024, docs.ultralytics.com/guides/yolo-performance-metrics/#class-wise-metrics.

Yeh, T.-W., S.-Y. Lin, H.-Y. Lin, S.-W. Chan, C.-T. Lin, and Y.-Y. Lin. "Traffic Light Detection Using Convolutional Neural Networks and Lidar Data." 2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Taipei, Taiwan, 2019, pp. 1-2. IEEE, https://doi.org/10.1109/ISPACS48206.2019.8986310.