Traffic Light Recognition with YOLOv8n

Aimee Oh Audrey Leong
Columbia Universtiy

1 Introduction

Autonomous vehicles have become an essential part of modern society, offering convenience and efficiency in getting from one place to another. After a slew of engineers and inventors contributing pieces of what eventually became the four wheeled autonomous vehicle, including Karl Benz's invention of the first practical autonomous vehicle and Nicolaus Otto's four-stroke internal combustion vehicles, many others have jumped into developing further technological advances for the car (Tho, 2024).

In the past decade, autonomous driving has become the forefront of this eagerness for advancing automobile technology. While some motivations may be because for the sake of advancement, researchers also push for autonomous driving to a). drive more safely and prevent car accidents and b). expand the ability to drive for more people, allowing more people to experience the convenience of driving.

According to the National Highway Safety Traffic Association, 94% of car accidents are caused by human error (Yurtsever et al., 2020), and according to the Federal Highway Administration, one quarter of traffic fatalities and half of all traffic injuries are due to intersections (USDoTFHA, 2024). Additionally, among fatal intersection crashes, drivers 85 or above were at most risk, followed by 20 years of age (Lombardi et al., 2017). Ages 55 and above are also overinvolved in fatal intersection crashes (Lombardi et al., 2017). With driving so heavily impacted by human error and demographics, the development of autonomous driving would benefit drivers and society greatly in hopefully eliminating human error and characteristics when driving and also allowing for older drivers to drive more safely, providing them with greater autonomy and ability to be on the road.

For autonomous driving to benefit drivers, especially with intersection crashes, they must be able to recognize traffic lights and classify them. In the autonomous driving industry, the problem of categorizing and recognizing traffic lights is called the Traffic Light Recognition (TLR). This is a challenge, more specifically because TLR must be successful in many different conditions (such as in rain or snow or at night) and must be extremely accurate – less than perfect would result in a fatal crash not unsimilar to one that occurs due to human error. Moreover, understanding traffic laws and applying general rules and context in addition to recognition is difficult and out of the scope for this current work. However, we recognize the importance of this in TLR and autonomous driving.

TLR is easy for drivers – when they are paying attention and are of good health, especially in eyesight. This important process while driving can be challenging and difficult for those who don't yet understand road rules, are easily distracted, or have bad eyesight. For these instances and to account for those who do fall into these circumstances, TLR gains importance in autonomous driving, as it is, arguably, the most fundamental way to decrease human error in fatal intersection crashes.

In this work, we use manually pre-processed data on traffic lights, collected from a dash-cam in California for TLR. Specifically, we train and fine-tune a YOLOv8-nano model with our data and classify traffic lights into three categories: red, yellow, and green. (summarized version of our method)

2 Related Works

In recent years, the most common method for TLR has been the use of Convolutional Neural Networks for TLR, which, as a general overview, is the use of convolution operations on input data

in order to find edges, textures, or shapes on the given image or video. Ren et. al uses Faster R-CNN, whose focus is the use of region proposals to detect objects (Ren et al., 2016). The Region Proposal Network (RPN) also used in Ren et. al. uses predictive bounding boxes and objectness scores to determine if a region has a high probability of containing an object, and only sending those candidates for further processing. RoI Pooling, a method of converting features from differently sized regions into a fixed-size grid, is then used to process the regions independently for future classification and regression calculation. This makes Faster R-CNN particularly good at detecting a multitude of various classes of objects in very busy images (Ren et al., 2016).

Within object tracking for TLR, Yang et. al.'s main method used is DeepSORT, which has been developed side by side with YOLO (Yang et al., 2022a). It is a deep learning method to SORT (Simple Online and Realtime Tracking) in order to track moving objects in video sequences. It enhances the original SORT algorithm, (a Kalman filter and Hungarian algorithm to measure correlation of frame-by-frame data), by extracting appearance features from the tracked target, rather than just depending on motion cues (Yang et al., 2022a). Because of this, it retains already identified objects, allowing occluded objects to remain tracked the moment they reappear, such as in traffic situations, where the constant shifting of trucks and cars may hide pedestrians and other cars temporarily. The Kalman filter afterwards predicts the future movement given the past movement, which also helps when detecting potentially occluded objects (Yang et al., 2022a).

Azimjonov et al. uses LSTM (Long Short-Term Memory Networks) to perform the predictions on these objects (Azimjonov et al., 2023). As a variant of an RNN (Recurrent Neural Network), they are geared for patterns that occur over a long period of time, due to their addition of memory cells (Alahi et al., 2016). These cells have three possible gates (input, forget, and output) to select what information to retain, which has proven useful in sequential data pattern detection. This helps prevent the issue that most RNNs have when learning data long-term, which is the continuously diminishing gradients during backpropagation. This leads to inaccurate predictions on patterns detected earlier on in the training, and rendering the long-term predictions inaccurate (Alahi et al., 2016).

However, most relevant to our work is the use of YOLO to train images to recognize traffic lights, also employed in Azimjonov et al. (Azimjonov et al., 2023) and Polley et al (Polley et al., 2024). Azimjonov et al. trains YOLOv3 for TLR and stacks a centroid based tracking method to link it to its nearest trajectory (Azimjonov et al., 2023). Polley et al. trains multiple YOLO models for traffic light detection and uses lane indicators to help with better TLR, including these as auxiliary indicators (Polley et al., 2024). In both these studies, YOLO (You Only Look Once) uses bounding boxes with assigned class probabilities, as it employs a grid based approach, where each cell gets processed. This allows object detection to be considered as a single regression problem (no intermediate step of generating region proposals), and instead each grid cell detects the images within it. The only additional step it requires is overlap detection, keeping only the highest confidence score box for each object. Not only does this improve the accuracy of the model, but also is fast (Redmon, 2016). This is particularly important when it comes to real-time computing in instances such as accident prevention and alerting authorities if need be.

class	Precision	Recall	Accuracy
All	0.61	0.63	0.52
0-Red	0.81	0.88	0.79
1-Yellow	0.79	0.77	0.77
2-Green	0.79	0.72	0.79

Figure 1: Polley et al. Precision, recall, and accuracy table (Polley et al., 2024)

3 Data

3.1 Data Collection

To collect data, we searched for traffic light datasets on Google Dataset Search, where we found Bosch traffic light data on Roboflow (karthik, 2023) and the SJTU Small Traffic Light Dataset (Yang et al., 2022b). The Bosch traffic data was taken and altered the images such that there were varying brightness levels for the same image. Additionally, this dataset only consisted of dash-cam images from the car in the US. We chose to continue with this data not only because it provided the greatest amount of images and data but also because the varying bright-

ness levels would allow for YOLOv8 to also train for different light conditions for TLR. the SJTU Small Traffic Light Dataset consisted of dash-cam and CCTV (stationary) images of traffic lights/intersections from China. The dash-cam images were of 720x80 resolution and the CCTV images were of 1080x1920 resolution, but we decided to combine these to get a more comprehensive and robust dataset to train the YOLOv8n model in. We chose to continue with this dataset because we wanted to see if the type of images and environment of the traffic lights would show differing results.

Dataset	Train	Test	Val
Bosch	3607	772	774
SJTUD	4050	868	868

Figure 2: SJTUD data Training Set precision, recall, and accuracy table (karthik, 2023)

3.2 Data Pre-processing

3.2.1 Bosch Dataset

The Bosch data contained 11 classes to categorize bounded objects: trafficLight, trafficLight-Red, trafficLight-Yellow, trafficLight-Green, trafficLight-RedLeft, trafficLight-GreenLeft, trafficLight-YellowLeft, car, truck, biker, and pedestrian. Among these classes, we decided to use only trafficLight-Red, trafficLight-Yellow, and trafficLight-Green to keep our TLR generalizable and applicable to different road laws, considering the varying traffic laws by state. Additionally, we did not consider classes car, truck, biker, and pedestrian because we were not aiming to train a model to predict whether an accident would happen or not. This was done in the process of converting the XML data format to YOLOv8 compatible format.

Taking each image and corresponding annotation file within the Bosch dataset, we located the coordinates and classification of each object that had been blocked in the image. Then, we only took the coordinates of the objects that were either trafficLight-Red, trafficLight-Yellow, or trafficLight-Green. These classes were further simplified to be 0, 1, and 2, respectively to normalize the classes and ensure consistency with the SJTU dataset that had different class names for the same classes (i.e. trafficLight-Green vs. green).

Figure 3: An image demonstrating a version of the image from the Bosch dataset.

Figure 4: An image demonstrating the a brightness augmented version of the same image.

3.2.2 STJUD China Dataset

For the STJUD Small Traffic Light Dataset, it contained 5 classes for the 1,080x1,920 images and 4 classes for 720x80 images. To make the model more generalizable, we also reduced these classes from wait, green, off, yellow, and red to red, yellow, and green. Similar to the Bosch Dataset, we took each image and corresponding file within the STJUD dataset (as it was also in XML format) and located the coordinations and classifications or each coordinates of the objects that were of each class. These classes were then again simplified to be 0, 1, and 2 (Red, Yellow, Green) to allow for consistency.

4 Methods and Comparison to Related Works

After reviewing the existing research on traffic light detection, and similar projects in the field such as traffic flow analysis, we have opted to use the You Only Look Once (YOLOv8) model

for our traffic light detection application. It is pretrained on the Common Objects in Context (COCO) dataset, which contains over 330K images annotated specifically for object detection, segmentation, and captioning tasks. Since the object categories already include objects like cars and bicycles, it seemed relevant to begin with the pretrained YOLOv8 model. In order to improve training speed and efficiency, while remaining within the GPU constraints capacity, we have implemented our traffic light detector on YOLOv8 nano, the smallest of the available models.

The approach for training and finetuning the model focused primarily on fixing the following parameters: image size, batch size, Intersection over Union (IoU) threshold, confidence threshold, and epoch size. Our second focus was on various forms of data augmentation due to the class imbalance that is inherent to the domain. Transformations, mosaic tiling, mixups, and HSV alterations were all considered for this task.

For the first dataset (STJUD China Dataset), we began by running the model with limited modifications. We opted for YOLO's default AdamW optimizer to adjust the learning rate, an image size of 480, a batch size of 8, and an initial run of 25 epochs, in order to see the GPU's limits. After running into several kernel interruptions, we have determined that both the image and batch sizes were too large if we were to train for a significant amount of epochs, and worked on decreasing the parameters to 340 and 4 respectively. Any smaller resulted in a loss of detail in the images, and too much noise throughout the training. Once this baseline was determined, we moved onto additional parameters, specifically the patience and detection parameters, which were set to 20 and 25 respectively, since we do not expect more than 25 object detections within a single frame, even with mosaic tiling and mixup augmentations.

While these parameters resulted in fairly high and stable precision and recall for the red and green classes, the main concern remained the yellow class, seeing as it contains considerably fewer samples within any dataset we found. Due to this, we began with the YOLO native augmentation parameters, adding mosaic at 0.5 and mixup at 0.2 in order to create additional samples where the objects would be found in different locations, hoping that it would also help the model generalize better in the end. This did little to improve the recall for the yellow class until we also added class weights, deemphasizing the red and green classes and augmenting the yellow class weights with the weight vector [0.25, 2, 0.25]. All of this combined with a lowered confidence threshold resulted in the highest accuracy we achieved with this model, at 70% during training and 68% overall accuracy during testing, with a total of 150 epochs trained.

To ensure that the high accuracy wasn't a result of overfitting, and to see if the low recall on the yellow dataset could be improved, we trained a second YOLOv8 model on a second dataset (BSTLD), by manually augmenting the data. Each photo has three copies, with augmentations such as horizontal flip, resizing the images to a standard 640px, rotation anywhere from an increased or decreased 15 degrees, and HSV brightness also increased or decreased by 15 percent. This was all done manually in the preprocessing step, before training it with similar parameters as the first: we kept the same batch size and image size, and checked the results before determining that the yellow class was still the weakest in terms of recall and accuracy. The same mosaic and mixup parameters as the first model were also applied, and the model was trained for 90 epochs in total, reaching a training accuracy 67.8% and a testing accuracy of 64.4%.

Figure 5: An image demonstrating the mosaic tiling augmentation of YOLOv8, collating 4 images to create new data.

5 Results and Analysis

5.1 Bosch Dataset

From our training of the YOLOv8 model on the Bosch dataset with three classes – Red, Yellow, and Green – we have a final overall test set accuracy of 0.679. While this accuracy level is not excellent, it is greater than chance. Moreover, if we take into account Polley et al.'s overall accuracy level for YOLOv8m trained with Bosch, we can see in Figure 6 that it is 0.52 for predicting different traffic lights correctly based on a given image. Based on this comparison, we can see that the overall accuracy level for our YOLOv8-nano model was greater than the related work.

class	Precision	Recall	Accuracy
All	0.758	0.684	0.664
0-Red	0.954	0.712	0.843
1-Yellow	0.981	0.591	0.790
2-Green	0.993	0.745	0.856

Figure 6: Bosch data Training Set precision, recall, and accuracy table (karthik, 2023)

class	Precision	Recall	Accuracy
All	0.74	0.705	0.679
0-Red	0.965	0.732	0.856
1-Yellow	0.848	0.650	0.782
2-Green	0.944	0.735	0.851

Figure 7: Bosch data Test Set precision, recall, and accuracy table (karthik, 2023)

Additionally, looking at Figures 6 and 7 for both the train and test set, we understand that the YOLOv8n model for the Bosch dataset performed best in detecting green lights for the test set and was worst at detecting yellow lights for both training and testing sets. This is unsurprising, as the dataset (and most datasets) contain the least data for yellow traffic lights, probably because they are harder to capture on a camera, as they only last for a few seconds, while green and red lights last longer. Moreover, given the implications of running a red light, the higher accuracy level for classifying a red light correctly is better than having a model that has a lower accuracy level for classifying a red light.

Other than accuracy levels, the overall test precision level is 0.74. Similar to accuracy levels, the precision level for Red is the highest at 0.967, with Green second best at 0.944 and Yellow at 0.848. This indicates that Red light predictions are the most likely to be correct while Yellow light predictions are least likely to be correct, which aligns with the overall trend of results.

The overall recall level is 0.499, with the recall level for Red at 0.732, Yellow at 0.650, and Green at 0.735. We can see that Red light predictions perform the best with Green closely following. Additionally, Yellow light predictions having the highest percentage of predictions to be false negatives is understandable as Yellow light prediction accuracy levels are the lowest and there is relatively very little Yellow light image data in the Bosch dataset – or any dataset in general.

From Figures 6 and 7, we can see that the test set actually performed better. This may be due to the regularization and early stopping we did when training the model. This probably allowed for the model to not be too biased towards the training data.

Overall, our YOLOv8-nano performed well, with a greater accuracy, precision, and recall rates than our related work Polley et al.

5.2 STJUD China Dataset

From our training of the YOLOv8n model on the STJUD China dataset with classes Red, Yellow, and Green, we have a final overall test set accuracy of 0.644. This test accuracy level is a little worse than the Bosch dataset. However, given that Polley et al's overall accuracy for YOLOv8m trained with Bosch is 0.52, our model on this dataset is better. Additionally, looking at Figure 8 and 9 for both the train and test set, we can see that the YOLOv8n model for the STJUDE dataset performed best in detecting red lights for both train and test set and was worst at detecting yellow lights, which is similar to the Bosch trained YOLOv8n model.

class	Precision	Recall	Accuracy
All	0.947	0.339	0.644
0-Red	0.899	0.335	0.613
1-Yellow	1.0	0.19	0.595
2-Green	0.942	0.492	0.724

Figure 8: SJTUD data Training Set precision, recall, and accuracy table (karthik, 2023)

class	Precision	Recall	Accuracy
All	0.74	0.705	0.679
0-Red	0.965	0.732	0.856
1-Yellow	0.848	0.650	0.782
2-Green	0.944	0.735	0.851

Figure 9: SJTUD data Test Set precision, recall, and accuracy table (karthik, 2023)

Other than accuracy levels, the overall test precision level is 0.947. Surprisingly, the precision level for Yellow is the highest at 1.0, with Red second best at 0.942 and Green at 0.899. This indicates that Yellow light predictions, when determined to not be a false negative, are the most likely to be correct, which actually is a change from other statistical results. The overall test recall level is 0.339, with the recall level for Red at 0.492, Yellow at 0.19, and Green at 0.335.

From Figures 8 and 9, we can see that the train set performed better than the test set, which is standard. Given that accuracy, precision, and recall levels are not as varying or far apart, we can infer that there is very little overfitting of the model on training data. While there may be some overfitting, it is not so much that the model is highly adapted to the noise in the training set.

Overall, our YOLOv8-nano on STJUD China dataset performed better than our related work Polley et al. but not better than our model trained on Bosch data. Not only are accuracy levels lower than the Bosch model statistics, but recall is significantly lower, meaning that there is a higher likelihood of false negatives in the STJUD China model than the Bosch model, meaning that it is a worse model.

6 Conclusion

This paper demonstrates the potential of using a YOLOv8-nano model for Traffic Light Recognition (TLR), a critical component of autonomous driving safety. Our research trained and fine-tuned the model on two distinct datasets: the Bosch dataset from the United States and the SJTU Small Traffic Light Dataset from China.

Despite being the smallest of the YOLOv8 models, YOLOv8-nano showed promising performance in detecting and classifying traffic lights, with overall test set accuracies of 0.679 for the Bosch dataset and 0.644 for the SJTU dataset. Notably, these accuracies surpass the baseline accuracy of 0.52 reported in related work by Polley et al.

Consistent across both datasets was the challenge of detecting yellow traffic lights. This difficulty likely stems from the inherent scarcity of yellow light images in training datasets, as yellow lights are transient and less frequently captured compared to red and green lights. However, the high precision for red light detection (0.965 in the Bosch dataset) still suggests the model's potential reliability in preventing potential intersection-related accidents.

Overall, the variations in performance between the Bosch and SJTU datasets underscore the importance of diverse and comprehensive training data. While the Bosch dataset yielded higher accuracy and recall, the SJTU dataset provided insights into the model's generalizability across different geographical and environmental contexts.

References

2024. The history of cars. Technical report, ThoughtCo.

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971.

Jahongir Azimjonov, Ahmet Özmen, and Metin Varan. 2023. A vision-based real-time traffic flow monitoring system for road intersections. *Multimedia Tools Appl.*, 82(16):25155–25174.

Mimansha Gupta, Harsha Miglani, Pradnyesh Deo, and Alka Barhatte. 2023. Real-time traffic control and monitoring. *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, 5:100211.

karthik. 2023. Autonomous driving dataset. `https://universe.roboflow.com/karthik-mc3w1/autonomous-driving-ng0z3`. Visited on 2024-12-09.

David A. Lombardi, William J. Horrey, and Theodore K. Courtney. 2017. Age-related differences in fatal intersection crashes in the united states. *Accident Analysis Prevention*, 99:20–29.

Nikolai Polley, Svetlana Pavlitska, Yacin Boualili, Patrick Rohrbeck, Paul Stiller, Ashok Bangaru, and J. Zöllner. 2024. Tld-ready: Traffic light detection – relevance estimation and deployment analysis.

J Redmon. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149.

USDoTFHA. 2024. About intersection safety. Technical report, U.S. Department of Transportation Federal Highway.

Feng Yang, Xingle Zhang, and Bo Liu. 2022a. Video object tracking based on yolov7 and deepsort. *arXiv preprint arXiv:2207.12202*.

Xue Yang, Junchi Yan, Wenlong Liao, Xiaokang Yang, Jin Tang, and Tao He. 2022b. Scrdet++: Detecting small, cluttered and rotated objects via instance-level feature denoising and rotation loss smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469.

7 Individual Contributions

Aimee Oh was responsible for the data collection, preprocessing, and training of the second model on the Bosch dataset. This includes the testing, finetuning, and interpretation of results. All of the writing in this report regarding that model, along with the introduction and conclusion are written by her. She has also contributed to the making and recording of the presentation.

Audrey Leong was responsible for the data collection, preprocessing, and training of the first model on the STJUD dataset. This includes the testing, finetuning, and interpretation of results. All of the writing in this report regarding that model, along with the related works and conclusion are written by her. She has also contributed to the making and recording of the presentation.