Report - Traffic Light Recognition

Mahesh Balasaheb Raut

Abstract

The traffic light recognition project aims to create deep learning neural network models utilizing the LISA Traffic Light dataset available on Kaggle, which can identify traffic lights in images. LISA is an acronym for "Learning from Imbalanced and Structured Aspects". The motivation behind this work is to develop accurate and efficient models that can be used in autonomous driving systems to detect traffic lights and enhance safety on the roads. The project utilized 9053 traffic light images from the total dataset captured in both daytime and nighttime, which were split into a training set (80%) and a testing set (20%). Two different state-of-the-art object detection models, Faster R-CNN and Single Shot MultiBox Detector (SSD), were trained on this dataset to detect traffic lights. Pre-trained neural networks from TensorFlow object detection were used for training. The data was prepared using TensorFlow object detection API scripts, and the models' performance was evaluated using mean Average Precision (mAP) and Recall. After 150,000 training steps, the Faster R-CNN model achieved an mAP of 27.4% and a recall of 13.44%, while the SSD model achieved an mAP of 30.97% and a recall of 14.31%. Overall, this project demonstrates the effectiveness of using deep neural networks for traffic light recognition.

1 Introduction

The problem of traffic light recognition is of great importance in the field of autonomous driving. Autonomous vehicles are being developed to increase road safety and reduce the number of accidents caused by driver error. A crucial component of an autonomous driving system is the ability to detect traffic lights accurately and efficiently.

This project introduces a comprehensive method for traffic light recognition that emphasizes the role of deep neural networks in constructing more accurate models for autonomous driving. The LISA Traffic Light dataset from Kaggle is a valuable resource for training deep neural networks for traffic light recognition. The dataset contains images captured in both daytime and nighttime sequences, which makes it suitable for developing models that can detect traffic lights in a variety of lighting conditions. The objective is to develop deep neural network models to detect traffic lights and compare the models.

1.1 Problem Definition

Traffic light recognition is a critical task for autonomous driving systems to ensure safe and efficient navigation on the road. Accurately detecting and classifying traffic lights is crucial for vehicles to make informed decisions on when to slow down, stop, or proceed through intersections.

Furthermore, some traffic lights may not be easily recognizable due to their orientation, distance, or occlusion by other objects. In some cases, traffic lights may be obstructed by trees, buildings, or other vehicles, making it even more challenging to detect and classify them accurately.

The problem can be formally defined as a supervised learning problem, where the models are trained on the labeled dataset to detect the presence of traffic lights in the images and accurately predict their state (green, yellow, or red) to support autonomous driving systems. To enhance the precision and robustness of traffic light recognition, cutting-edge computer vision and deep learning methods, including object detection algorithms and convolutional neural networks, will be utilized in this project. Specifically, Faster R-CNN and SSD will be employed to achieve this goal. The models are trained to output the bounding box coordinates of any traffic lights present in the input image.

The problem of traffic light recognition is important for the development of autonomous driving systems and has potential applications in other areas such as traffic management and surveillance.

Deep neural networks are a promising approach to address these challenges and developing accurate and efficient traffic light recognition models.

1.2 Dataset Characteristics

A total of 9053 images of traffic lights captured during both daytime and nighttime sequences, were utilized for this project. Specifically, the dataset comprises two sets of images, referred to as day sequence and night sequence, which contain 4060 and 4993 images, respectively.

To create a comprehensive dataset for traffic light recognition, the frames from both sequences were combined, resulting in a total of 9053 images. In addition, the corresponding annotation files of the day sequence and night sequence were also combined. The dataset includes annotations of bounding boxes around traffic lights, which indicate their class in the image. The dataset's annotation file consists of three classes, namely stop, go, and warning.

The following are examples of images from both the day and night sequences, each with corresponding annotated labels:

Figure 1: Day Sequence - 00244.jpg

Figure 2: Day Sequence - 01960.jpg

Exploratory analysis of the data revealed that the day sequence set includes 3348 images containing

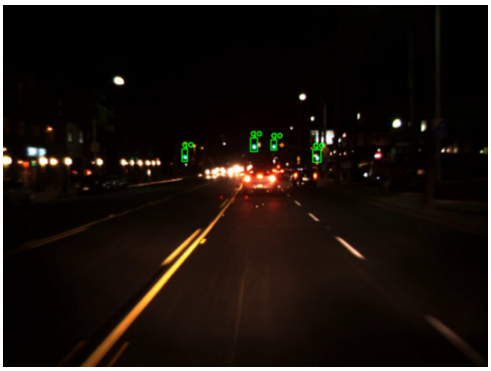

Figure 3: Night Sequence - 04583.jpg

Figure 4: Night Sequence - 01452.jpg

traffic lights and 712 images without traffic lights, whereas the night sequence set comprises 4760 images with traffic lights and 233 images without traffic lights. The images have an average size of 1280 x 960 pixels and are color images with a depth of 3 channels.

2 Related Work

The problem of traffic light recognition has been studied extensively in recent years due to its importance in the development of autonomous driving systems. Several deep learning-based approaches have been proposed to address this problem.

The paper (Gupta and Choudhary, 2019) proposes a camera-based framework for automated traffic light detection and recognition using Faster R-CNN and Grassmann manifold learning. Transfer learning on VGG16 is used for feature extraction and creating subspaces for each traffic light color. Discriminant analysis on the manifold is employed for recognition and experiments show high accuracy and robustness compared to state-of-the-art methods.

(Wang et al., 2021) proposed an improved YOLOv4 algorithm to enhance the detection and recognition precision of traffic lights. This is

achieved by using a shallow feature enhancement mechanism and a bounding box uncertainty prediction mechanism. The algorithm is evaluated on the LISA traffic light data set and is shown to have a high effectiveness in enhancing the detection and recognition precision of traffic lights, with a significant improvement in the area under the PR curve and mean average precision compared to the original YOLOv4 algorithm. The improved YOLOv4 algorithm is considered a robust and practical method for use in the recognition of traffic signal lights.

(Zeng et al., 2023) study proposed a traffic light recognition algorithm that uses YOLOv5s object detection algorithm for traffic light detection and SVM classifier for state classification. The proposed network structure called N_ResNet is capable of detecting traffic lights of different sizes, especially medium-sized traffic lights. The results show that the proposed network achieves a significant improvement in recall compared to DarkNet. The overall recall is improved by 2.66%, and the accuracy of the SVM classifier reaches 92.2%.

This paper (Che et al., 2020) presents a two-stage approach for traffic light recognition based on image processing and deep learning. It considers accuracy, runtime, and size and uses perspective relationship and fractal dimension in detection, and SqueezeNet for classification. The proposed approach is competitive in terms of accuracy and runtime, and suitable for implementation on smart, mobile or embedded devices. (Possatti et al., 2019) work puts forward a system for autonomous cars to recognize relevant traffic lights using a combination of deep learning-based detection and prior maps. The system is evaluated on five test cases in the city of Vitória, showing promising results in correctly identifying relevant traffic lights along the trajectory. The proposed approach can potentially improve the performance and safety of autonomous driving.

(Bach et al., 2018) describes a unified deep convolutional traffic light recognition system that was developed to detect and classify traffic lights, as well as distinguish their type. The system is based on the Faster R-CNN architecture and was evaluated on the DriveU Traffic Light Dataset, achieving an overall detection performance of 0.92 Average Precision for traffic lights of width greater than 8 pixels. The study identified pedestrian lights as the main cause of false positives, and inconsistencies were revealed among multiple detections in single images when assessing the traffic light states for all present driving directions.

3 Methodology

To prepare the data, the TensorFlow 2 Object Detection API scripts was used to convert the raw data into a format compatible with the object detection models.

3.1 Deep Learning for Object Detection

The reason for utilizing Deep Learning is that it directly addresses the challenges associated with the traffic light recognition by:

1. "Learning" very distinguishing features which improves accuracy.
2. Sharing computation when computing these features, which improves speed.
3. Learning enough features distributions to be able to generalize well on new unseen images.

There are two types of Deep Neural Networks used for Traffic Light Recognition:

1. "One stage" object detectors (SSD).
2. "Two stages" object detectors (Faster R-CNN).

The algorithms used in this project, Faster R-CNN and SSD, are well-established in the field of object detection and have been shown to achieve state-of-the-art performance on a range of datasets.

3.2 Software Setup

This project involves the creation of a new environment on a local machine, named data6400, with a Quadro M1200 Nvidia GPU consisting of 4GB. The machine uses CUDA version 11.7.0 and cuDNN version 11.5.0, which is a CUDA library for deep learning. The installed version of TensorFlow is 2.9.0.

To use the TensorFlow 2 Object Detection API, the Protobuf libraries are required to configure the model and training parameters. Additionally, the pycocotools package is listed as a dependency for the Object Detection API. The process of installing the Object Detection API involves installing the corresponding package. This can be done by executing certain commands from within the tensorflow-models-research directory, as explained in the source (Vladimirov, 2020).

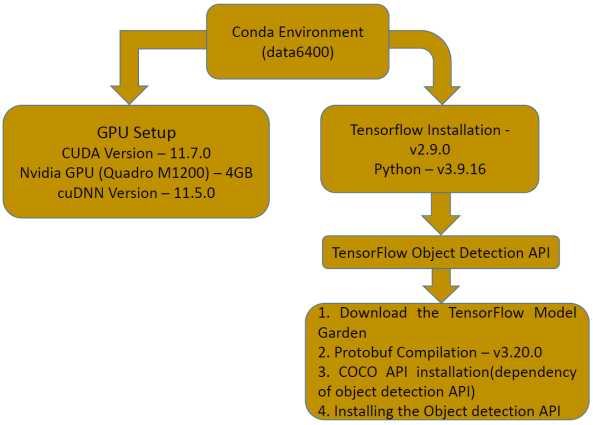

Figure 5: Software Setup Flow Diagram

3.3 Faster R-CNN Overview

Faster R-CNN is a well-known object detection algorithm that improves on its predecessors, such as R-CNN and Fast R-CNN. It introduces a Region Proposal Network (RPN) that allows the detection network to share convolutional features and efficiently propose object regions in an image. This feature eliminates the requirement for external region proposal techniques, resulting in a faster algorithm than its predecessors.

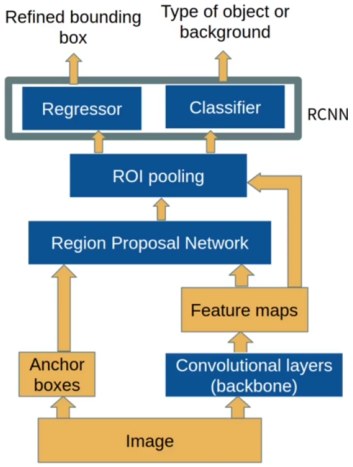

Figure 6: Faster R-CNN Block Diagram

The steps involved in the Faster R-CNN algorithm are:

1. Input image is passed through a convolutional neural network (CNN) to generate a feature map.

2. A Region Proposal Network (RPN) is applied to the feature map to generate object proposals. The RPN uses a sliding window approach to propose regions of interest, which are scored based on how well they overlap with ground-truth objects in the training data.

3. The proposed regions are then refined using a bounding box regression network. The refined regions are called ROIs (Region of Interest).

4. Each ROI is passed through a classification network to determine whether it contains an object and, if so, what class it belongs to.

5. The final output is a set of object detections, each represented by a bounding box and a class label.

Overall, the Faster R-CNN algorithm combines both region proposal and object detection in a single unified architecture, making it much faster and more accurate than previous approaches.

3.4 Single Shot MultiBox Detector (SSD) Overview

Single Shot MultiBox Detector (SSD) is a deep learning-based object detection algorithm that uses convolutional feature maps of different resolutions to detect objects of varying scales and aspect ratios in a single pass. Unlike traditional object detection algorithms that use a two-stage approach, SSD can directly predict object classes and bounding boxes in a single forward pass, making it much faster.

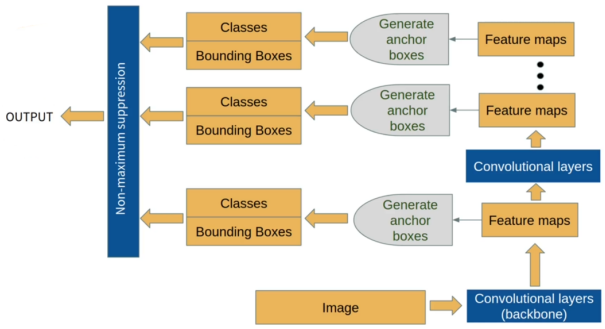

Figure 7: SSD Block Diagram

Here are the main steps in Single Shot MultiBox Detector (SSD):

1. Input images are passed through a base convolutional network (such as VGG or ResNet) to extract feature maps at different scales.

2. At each scale, a set of default bounding boxes, or anchor boxes, are defined based on different aspect ratios and scales.

3. For each default box, the network predicts the class probabilities and the offsets for the predicted box relative to the default box.

4. The predictions from all scales are combined, and non-maximum suppression is applied to remove redundant detections.

5. The final output consists of a set of bounding boxes and their corresponding class labels and confidence scores.

By using a single deep neural network to predict object classes and bounding boxes in a single pass, SSD achieves high speed and accuracy in object detection.

3.5 Selecting the right model architecture

When selecting a traffic light recognition model architecture, accuracy and time are crucial factors to consider. Accuracy refers to the precision of the neural network's predictions, while time represents the duration it takes to make a prediction.

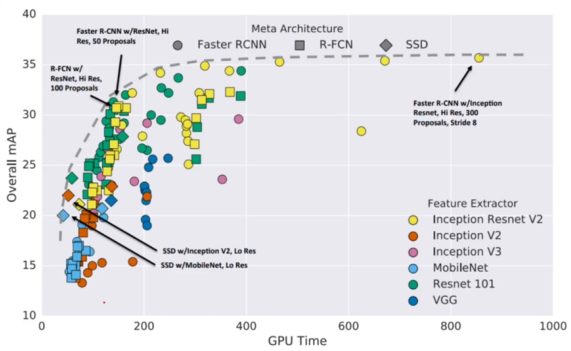

Figure 8: Accuracy Vs Time

In a referenced paper (Huang et al., 2017), the authors conducted numerous experiments on three types of neural networks, namely Faster RCNN, RFCN, and SSD. The graph shows the inference time on the X-axis and mean average precision on the Y-axis. Faster RCNN-based neural networks are the most accurate, as represented by the circles, while RFCN tends to have lower accuracy. The graph is useful in making decisions about using SSD, Faster RCNN, or other models since it provides information about accuracy and time.

3.6 Data Preparation

It should be noted that the TensorFlow 2 Object Detection API was utilized for this project. The annotated files for both the day sequence and night sequence were merged into a single annotation file. During data exploration, it was discovered that the annotated labels consisted of stop-12875, go-12558 (representing green), and warning-981.

To create the train and test sets, the images from day sequence and night sequence were combined and split into 80% train i.e. 7243 images and remaining 20% test i.e. 1810 images. A total of 26,414 annotated labels were provided with the dataset, and they were annotated by humans to ensure accuracy and reliability. After random splitting, the training set contains 21,157 annotated labels, and the testing set contains 5,257. annotations_train and annotations_test csv files were created to reflect the train-test split. The labelmap.pbtxt file contained information about the three classes present in the dataset: stop, go, and warning.

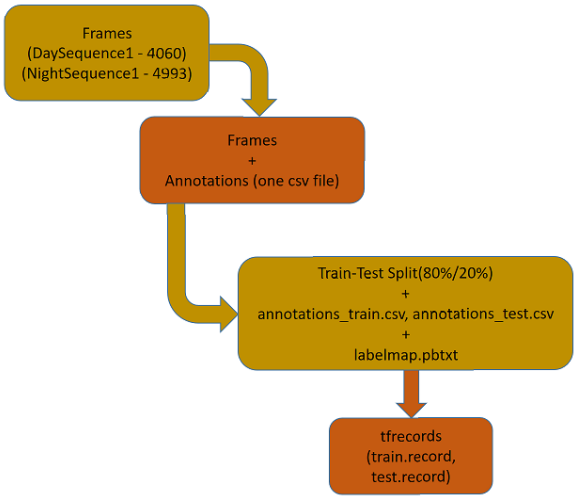

Figure 9: Tensorflow Object Detection Data Preparation

For TensorFlow object detection, train.records and test.records were created, which contained lists of annotations for the dataset images. TFRecords is a binary file format used in TensorFlow for efficiently storing and reading large datasets. It is a way of storing data in a serialized format that can be easily parsed by TensorFlow during training. This format allows the data to be compressed, and also enables TensorFlow to read the data in parallel, which can speed up training.

To train the models, the TensorFlow 2 Object Detection API was used and pre-trained weights from the COCO dataset. Steps taken for building an object detector using Tensorflow 2 object detection API:

1. The pre-trained neural networks for Faster R-CNN and SSD resnet50 were downloaded.

2. The configuration file corresponding to each pre-trained neural network was downloaded.

3. The configuration files were modified based on specific requirements, such as setting the path to train.records, path to labelmap.pbtxt, and path to test.records.

4. The training and evaluation of the neural networks were performed using TensorFlow 2 API scripts.

The models were fine-tuned on the dataset using a learning rate of 0.001 for 150,000 steps, with a warmup period of 5000 steps. The batch size was limited to 1 due to GPU size constraints.

3.7 Directory Structure

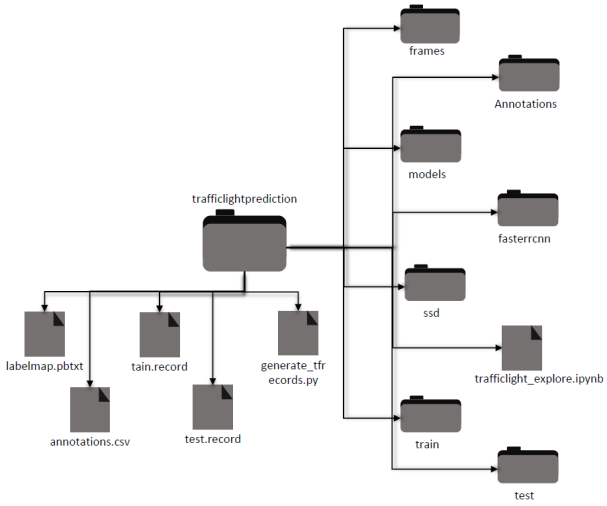

Figure 10: Workplace Directories

The main directory named "trafficlightprediction" includes several important files and directories. The "generate-tfrecords.py" file in the main directory is used to generate "train.record" and "test.record" files. In addition, the main directory contains "annotations.csv" and "labelmap.pbtxt" files, while the "frames" directory stores all the images of the day and night sequences. The "models" directory includes script files for TensorFlow's object detection API, and the "fasterrcnn" and "ssd" directories contain pre-trained models to be used as starting points for training jobs. The "trafficlight-explore.ipynb" file is utilized for data exploration, and the "trafficlight-dataset.ipynb" file is used to split the images into training and testing sets, which are saved in the "train" and "test" directories, respectively.

3.8 Training of models

3.8.1 Training with Faster R-CNN

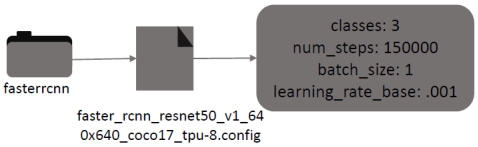

Figure 11: Faster R-CNN config

- To train Faster R-CNN on local:

```
# From the tensorflow/models/research/ directory
python object_detection/model_main_tf2.py --pipeline_config_path
trafficlightprediction/fasterrcnn/faster_rcnn_resnet50_v1_640x640_coco17_tpu-8.config
--model_dir trafficlightprediction/fasterrcnn/training_process/ --alsologtostderr
```

- To test/eval Faster R-CNN on local:

```
# From the tensorflow/models/research/ directory
python object_detection/model_main_tf2.py --pipeline_config_path
trafficlightprediction/fasterrcnn/faster_rcnn_resnet50_v1_640x640_coco17_tpu-8.config
--model_dir trafficlightprediction/fasterrcnn/training_process/ --checkpoint_dir
trafficlightprediction/fasterrcnn/training_process/ --alsologtostderr
```

3.8.2 Training with SSD

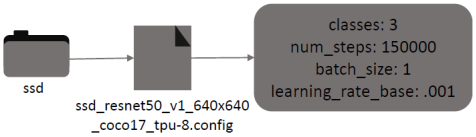

Figure 12: SSD config

- To train SSD on local:

```
# From the tensorflow/models/research/ directory
python object_detection/model_main_tf2.py --pipeline_config_path
trafficlightprediction/ssd/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.config
--model_dir trafficlightprediction/ssd/training_process/ --alsologtostderr
```

- To test/eval SSD on local:

```
# From the tensorflow/models/research/ directory
python object_detection/model_main_tf2.py --pipeline_config_path
trafficlightprediction/ssd/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.config
--model_dir trafficlightprediction/ssd/training_process/ --checkpoint_dir
trafficlightprediction/ssd/training_process/ --alsologtostderr
```

4 Evaluation

In accordance with standard evaluation procedures, the testing dataset was comprised of 20% of the total data, specifically 1635 frames. The models were

assessed using various metrics, including mean Average Precision (mAP) and Recall, which were calculated using the object detection capabilities provided by TensorFlow and visualized on the TensorBoard dashboard. Additionally, both models were scrutinized for their respective losses.

During the evaluation process, the evaluation script was used, provided by the TensorFlow 2 object detection API. This script calculates the mAP and recall of the models on the test set using Intersection over Union (IoU) thresholds of 0.5, 0.75, and 0.95.

4.1 Faster R-CNN Results

4.1.1 Faster R-CNN Detection Boxes Precision

At 150,000 steps, the faster r-cnn model achieved a mean average precision (mAP) of 0.274, calculated by averaging over IoU thresholds ranging from 0.5 to 0.95. Additionally, the mAP was reported at specific IOU thresholds of 0.50 and 0.75, which were 0.5421 and 0.2577, respectively.

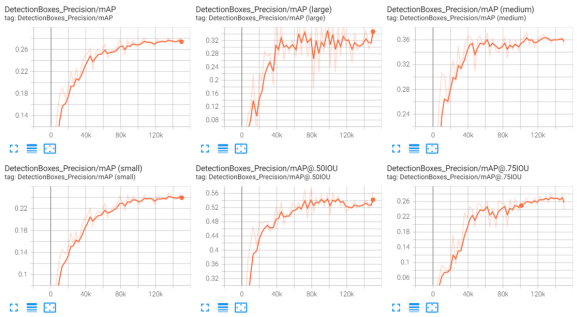

Figure 13: Faster R-CNN mAP graph

Furthermore, the model achieved mean average precision of 0.2393 for small objects (with an area less than 32^2 pixels), mean average precision of 0.3575 for medium-sized objects (with an area between 32^2 pixels and 96^2 pixels), and mean average precision of 0.3473 for large objects (with an area between 96^2 pixels and 10000^2 pixels).

4.1.2 Faster R-CNN Detection Boxes Recall

The Faster R-CNN model achieved an average recall rate of 0.1344 at 150,000 training steps with only one detection (AR@1), and the average recall rate increased to 0.3614 and 0.3814 with 10 and 100 detections, respectively (AR@10 and AR@100). The average recall rate for small objects was 0.3385 with 100 detections, while for medium and large objects, it was 0.4892 and 0.4824, respectively. Overall, the model exhibited better recall rates for medium and large objects compared to small objects.

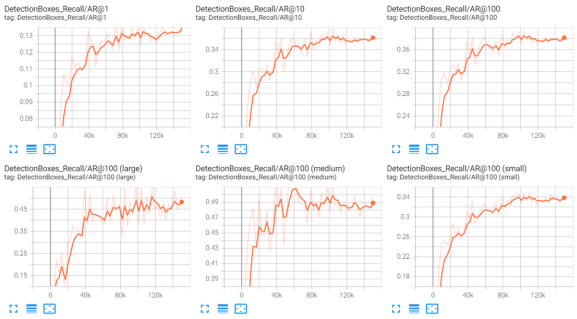

Figure 14: Faster R-CNN recall

4.1.3 Faster R-CNN Losses

The Faster R-CNN model had a total loss of 0.1223 after 150,000 training steps, which includes losses from different components of the model. The losses for the Region Proposal Network are reported separately, with a localization loss of 0.0191 and an objectness loss of 0.006794. The final classifier losses are also reported, including a classification loss of 0.04605 for three classes and a localization loss of 0.05035 for the bounding box regressor.

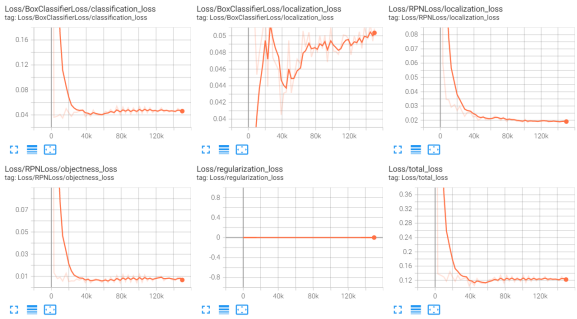

Figure 15: Faster R-CNN Losses

4.2 SSD Results

4.2.1 SSD Detection Boxes Precision

The SSD object detection model achieved a mean average precision (mAP) of 0.3097 after 150,000 training steps. At 50% IOU, the mAP was 0.5976 (called mAP@.50IOU), and at 75% IOU, the mAP was 0.2874 (called mAP@.75IOU).

The model's evaluation metrics for object detection include mean average precision values for small, medium, and large objects. The values are

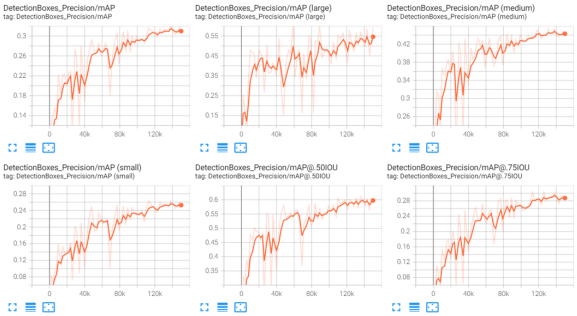

Figure 16: SSD mAP

0.2527 for small objects (area < 32^2 pixels), 0.4428 for medium-sized objects (32^2 pixels < area < 96^2 pixels), and 0.5462 for large objects (96^2 pixels < area < 10000^2 pixels).

4.2.2 SSD Detection Boxes Recall

The object detection performance of SSD model at 150,000 training steps is evaluated based on three different levels of average recall: AR@1, AR@10, and AR@100. The values of AR@1, AR@10, and AR@100 are 0.1431, 0.4063, and 0.4317, respectively.

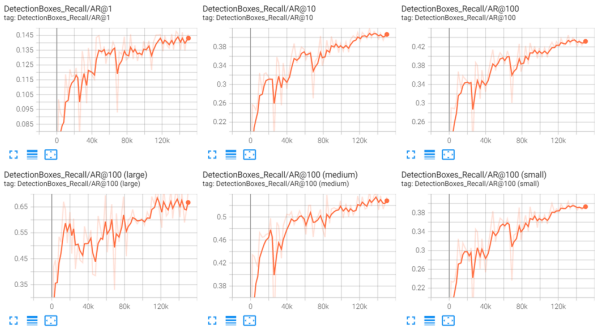

Figure 17: SSD recall

Additionally, the average recall for small objects with 100 detections is 0.393, for medium objects it is 0.5282, and for large objects it is 0.6672. These results are reported in terms of the 'DetectionBoxes_Recall' metric.

4.2.3 SSD Losses

The loss components of an SSD object detection model were evaluated at 150,000 training steps, with the following results reported:

1. Loss/classification_loss: 0.3593
2. Loss/localization_loss: 0.2461
3. Loss/regularization_loss: 0.1764
4. Loss/total_loss: 0.7817

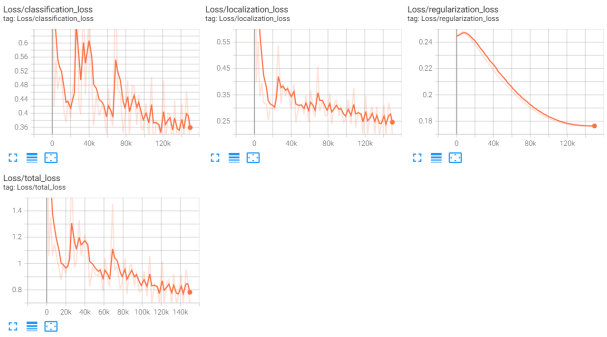

Figure 18: SSD Losses

These loss values provide an indication of the model's performance during training, with classification_loss and localization_loss ideally decreasing as the model improves its ability to classify and localize objects, and the regularization_loss helping to prevent overfitting. The total_loss represents the combined loss of all components, and is used to update the model's parameters during training.

4.3 Comparison of Faster RCNN and SSD

The table 1 presented compares two object detection models, Faster R-CNN and SSD, which were both pretrained on the resnet50 architecture and trained for 150K steps with a batch size of 1. The models were trained on a Quadro M1200 with 4GB memory. The training time for Faster R-CNN was 1 day, 10 hours, 48 minutes, and 21 seconds, while the training time for SSD was 1 day, 15 hours, 44 minutes, and 19 seconds. The results show that SSD has higher classification and localization loss, but achieves a higher mAP and recall than Faster R-CNN.

5 Conclusion

The project aimed to develop and train deep neural network models using the LISA Traffic Light dataset from Kaggle, and evaluate their performance using metrics such as mean Average Precision and Recall. The results indicated reasonable accuracy for traffic light recognition using both Faster R-CNN and SSD models. Specifically, the Faster R-CNN model achieved an mAP of 27.4% and a recall of 13.44%, while the SSD model achieved an mAP of 30.97% and a recall of 14.31%.

	Faster R-CNN	SSD
Pretrained Model	faster_rcnn_resnet50_v1	ssd_resnet50_v1
Steps Trained	150K steps	150K steps
Batch Size	1	1
Training Time	1d 10h 48m 21s	1d 15h 44m 19s
Trained On	(Quadro M1200) – 4GB	(Quadro M1200) – 4GB
Classification_loss	0.04605	0.3593
Localization_loss	0.05035	0.2461
Total_loss	0.1223	0.7817
mAP	0.274	0.3091
Recall	0.1344	0.1431

Table 1: Comparison of Faster RCNN and SSD models

Although these results are promising, there are limitations and areas for improvement. One limitation is the relatively small size of the training dataset, which may cause overfitting and limit the generalizability of the models to other scenarios. Additionally, only two state-of-the-art object detection models were utilized, and there may be other models or variations that could achieve better performance.

Future work could involve collecting and utilizing a more diverse and larger dataset to train the models. Additionally, exploring different object detection models or variations such as YOLO or Mask R-CNN could help in comparing their performance for traffic light recognition.

References

Martin Bach, Daniel Stumper, and Klaus Dietmayer. 2018. Deep convolutional traffic light recognition for automated driving. *21st International Conference on Intelligent Transportation Systems (ITSC)*.

Michael Börnö, Jakob Scheible, Nicolas Müller, and Markus Enzweiler. 2016. Lisa traffic light dataset. https://www.kaggle.com/datasets/mbornoe/lisa-traffic-light-dataset.

Mingliang Che, Mingjun Che, Zhenhua Chao, and Xinliang Cao. 2020. Traffic light recognition for real scenes based on image processing and deep learning. *COMPUTING AND INFORMATICS*, 39(3):439–463.

Any Gupta and Ayesha Choudhary. 2019. A framework for traffic light detection and recognition using deep learning and grassmann manifolds. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 600–605.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. 2017. Speed/accuracy trade-offs for modern convolutional object detectors.

Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. 2016. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1800–1815.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. 2015. SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Mark Philip Philipsen, Morten Bornø Jensen, Andreas Møgelmose, Thomas B. Moeslund, and Mohan M. Trivedi. 2015. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2341–2345.

Lucas C. Possatti, Rânik Guidolini, Vinicius B. Cardoso, Rodrigo F. Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. 2019. Traffic light recognition using deep learning and prior maps for autonomous cars. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Lyudmil Vladimirov. 2020. Tensorflow object detection api installation. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html.

Qian Wang, Qian Zhang, Xuecheng Liang, Yuqi Wang, Chunlei Zhou, and Vitaly Ivanovich Mikulovich. 2021. Traffic lights detection and recognition method based on the improved yolov4 algorithm. *Sensors (Basel)*, 22(1):200.

Shengkun Zeng, Ran Wang, Mengshan Li, and et al. 2023. N_resnet: A real-time traffic light recognition network using object detection. *PREPRINT (Version 1) available at Research Square*.