CSC51073EP Image Analysis and Computer Vision Report
Group p22

Yangtao Fang, Taiwei Wu

December, 2024

1 Introduction

The authors hail from China, a country with a vast territory but a staggering population of 1.4 billion. A significant portion of the western and northern regions of China is not suitable for human habitation, resulting in 96% of the population residing in 36% of the land in the southeast [19]. This leads to the majority of Chinese living in crowded cities. The large urban population also owns a tremendous number of vehicles. For instance, in 2023, Shenzhen, the hometown of one of the authors, had 4.1726 million vehicles [2], while the city's area is only 1,997 square kilometers, less than one-sixth the size of Île-de-France. Consequently, traffic congestion is common on Chinese city roads. One of the most famous traffic jams was the China National Highway 110 traffic jam, which lasted 12 days, with thousands of vehicles stretching over 100 kilometers. The "spectacle" of this traffic jam is shown in Figure 1.

Figure 1: China National Highway 110 traffic jam

We hope to combine computer vision methods and do our part to alleviate traffic congestion. Therefore, we plan to develop a traffic flow monitoring system that conducts real-time road traffic

monitoring. When the traffic flow on the road exceeds a certain threshold, the system will issue an early warning, reminding drivers to avoid this road as much as possible when traveling, thereby preventing vehicle clustering and avoiding congestion to a certain extent.

Furthermore, considering that vehicle speeds vary under different weather conditions, such as when drivers choose to drive slowly due to slippery roads when it rains or snows, or when drivers reduce their speed due to low visibility in hazy or foggy weather, our monitoring system also includes a real-time weather monitoring module. This allows for setting different thresholds according to the weather, better avoiding congestion.

We worked iteratively on different versions of our models to try to compare the relevance of different features that we could identify throughout the project.

1.1 Project Background

The main focuses of this project are detecting vehicles appearing in videos and weather recognition. For vehicle detection, we learned that CNN-based deep learning algorithms have an absolute advantage in vehicle detection in machine vision [17]. Theoretical research related to deep learning first appeared in the 1950s. Early deep learning theories were very immature, and model training lacked sufficient data and hardware support. Therefore, training high-performance models without overfitting was a challenging task. With the advent of computer hardware and large-scale datasets, deep learning has achieved unprecedented development. In particular, deep learning algorithms based on Convolutional Neural Networks (CNNs) have achieved tremendous success in multiple machine vision tasks [8, 9]. Since then, CNN-based deep learning object detection models have been rapidly evolving. Specifically, deep learning can automatically learn vehicle detection features from training samples using data-driven methods. This process does not require prior knowledge or manual feature design, making vehicle feature extraction simpler, more objective, and richer. The powerful feature extraction capabilities enable deep learning algorithms to easily handle vehicle detection tasks in various traffic scenarios. The accuracy and robustness of vehicle detection are also superior to traditional methods and machine learning algorithms. Therefore, we chose to use deep learning-based machine vision vehicle detection methods.

According to the principles of the algorithms and the process of vehicle detection, deep learning-based vehicle detection methods can be divided into two categories: two-stage detection algorithms and one-stage detection algorithms [15]. Two-stage detection algorithms divide the vehicle detection task into two stages: generating vehicle region proposals and finding vehicle targets from the region proposals. Typical representatives of two-stage methods include R-CNN [3] and SPP-Net [4]. One-stage detection algorithms eliminate the operation of generating vehicle region proposals and unify vehicle recognition and detection into a single network for processing. Typical representatives of one-stage methods include the YOLO series models [13], SSD [10], and M2Det [23]. The accuracy of two-stage detection methods is generally higher than that of one-stage detection methods. Moreover, due to the candidate region generation strategy, this type of model has a good effect on detecting small vehicle targets. However, one-stage detection methods have an absolute advantage in real-time performance, which is crucial for intelligent vehicles. With the continuous improvement of one-stage methods, they have gradually overcome the problems of low detection accuracy and small targets, becoming the mainstream deep learning-based machine vision vehicle detection methods.

For weather recognition, one approach is using outdoor images is to classify the images based on atmospheric optical features [16]. Typically, we calculate features based on some prior knowledge and then use classifiers on these feature sets. For example, for fog recognition, Pavlic [12] extracted global descriptors of the image through Gabor transform and achieved fog detection; for rain recognition, Zhang [22] successfully identified rain by utilizing the motion and color characteristics of raindrops. For snow recognition, Xu [20] obtained good snow recognition results by introducing a luminance model and a dynamics model. These traditional recognition methods, which rely on manually extracted features, have a smaller computational load and can achieve good recognition results for specific types of weather phenomena through well-established models. However, their drawbacks include lower accuracy and narrower applicability [14]. After constructing the model, it is difficult to improve it. Moreover, due to the rapid development of the deep learning field in recent years, it has also promoted various aspects of computer vision. In particular, the application of Convolutional Neural Networks (CNNs) has enabled people to access deep semantic information of images that was previously inaccessible [9], which has greatly improved the accuracy of image clas-

sification. Furthermore, if we use CNNs to build a classification model for visual weather images, the recognition speed will be faster due to their characteristics of local perception and parameter sharing. Therefore, we also plan to use deep learning-based models for weather recognition.

1.2 Project Objectives

The objective of this project is to develop a vehicle detection model that can accurately identify the number of vehicles on the road in real time. This model will then be used to issue an early warning when the traffic flow exceeds a pre-defined threshold. Furthermore, a weather detection module is required to ascertain the weather conditions on the road in real time. This will facilitate the dynamic adjustment of the warning threshold in accordance with the weather on the road. Additionally, a GUI (Graphical User Interface) should be developed to ensure user-friendly operation and facilitate model parameter adjustment.

1.3 Vehicle Dataset

Firstly, we selected the Road Vehicle Images Dataset - Bangladeshi road vehicle images with YOLO v5 annotation for our experiments [21]. This dataset contains 21 vehicle label classes: 'ambulance', 'army vehicle', 'auto rickshaw', 'bicycle', 'bus', 'car', 'garbagevan', 'human hauler', 'minibus', 'minivan', 'motorbike', 'pickup', 'policecar', 'rickshaw', 'scooter', 'suv', 'taxi', 'three wheelers -CNG-', 'truck', 'van', and 'wheelbarrow'. However, the training set consists of only 2,704 images, which is too small in scale. We attempted to train the YOLOv5 model on this dataset, but the results were extremely poor, as the average F1 score for all labels is less than 0.5 at any confidence level, as shown in Figure 2.

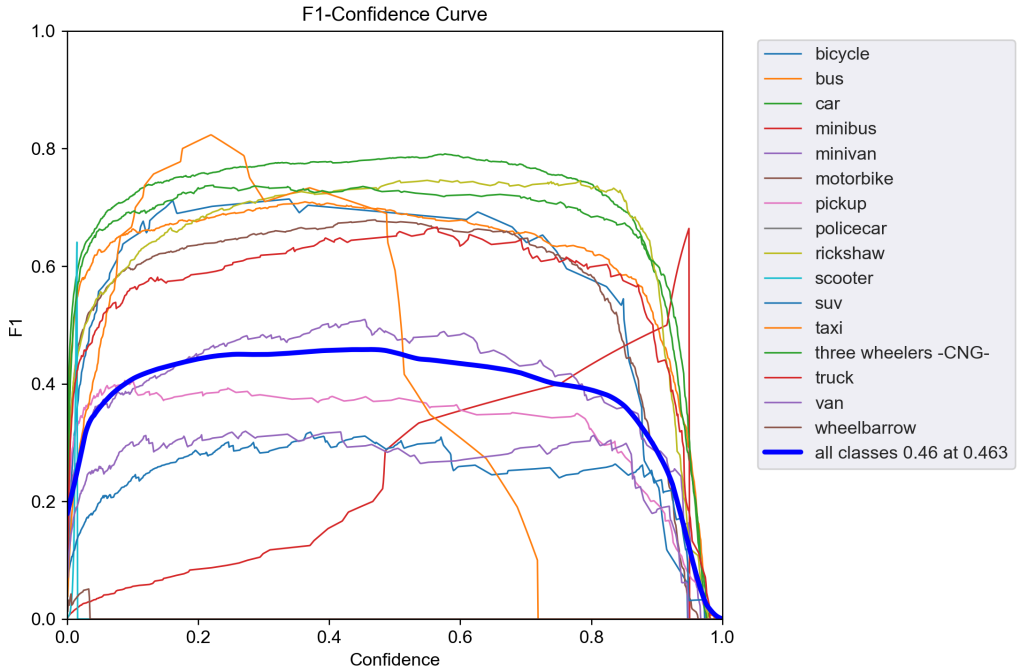

Figure 2: F1-Confidence curve of Road Vehicle Images Dataset.

We finally decided to use UA-DETRAC dataset to train our vehicle detection model [18]. The dataset consists of 10 hours of videos captured with a Cannon EOS 550D camera at 24 different locations at Beijing and Tianjin, two of the busiest cities in China. The videos are recorded at 25 frames per seconds, with resolution of 960 x 540 pixels. The dataset comprises 140,000+ frames, which have been annotated for 8,250 vehicles. In total, 1.21 million vehicle bounding boxes have been annotated. The annotation process was conducted by more than ten domain experts over a period of more than two months. The vehicles have been categorised into four groups: car, bus, van and other vehicles (including other vehicle types such as trucks and tankers). The style of the UA-DETRAC record is shown in Figure 3.

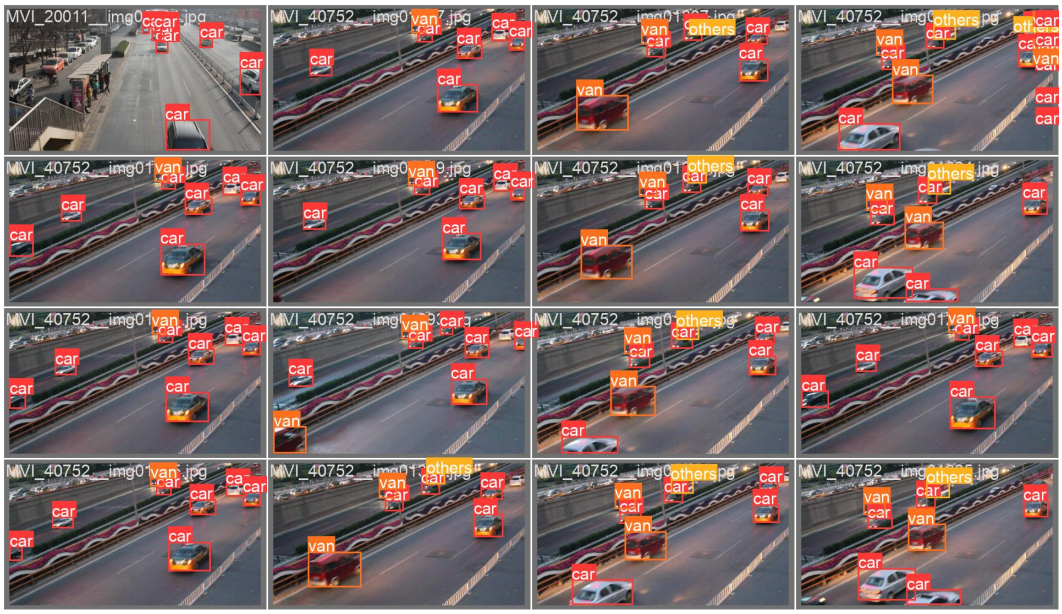

Figure 3: Example of UA-DETRAC dataset

2 Model Choice

2.1 YOLOv5

YOLO (You Only Look Once) is a popular object detection and image segmentation model [13]. Introduced in 2015, YOLO quickly gained popularity due to its high speed and accuracy. YOLOv2, released in 2016, improved upon the original model by incorporating batch normalization, anchor boxes, and dimension clustering. YOLOv3, launched in 2018, further enhanced the model's performance by using a more efficient backbone network, multiple anchors, and spatial pyramid pooling. Released in 2020, YOLOv4 introduced innovative techniques such as Mosaic data augmentation, a new anchor-free detection head, and a new loss function. YOLOv5 further improved the model's performance and added new features like hyperparameter optimization, integrated experiment tracking, and automatic exporting to common export formats. Currently, the YOLO series has progressed to YOLOv11, but due to the newer models having many features that we do not require, we ultimately chose YOLOv5 for our project to balance the model's training cost and performance [6].

2.2 ResNet-50

ResNet-50 is a CNN architecture that belongs to the ResNet (Residual Networks) family [5], a series of models designed to address the challenges faced when training deep neural networks. ResNet-50 is renowned for its depth and efficiency in image classification tasks. The ResNet architecture comes in various depths, such as ResNet-18, ResNet-32, ResNet-101, etc., with ResNet-50 being a medium-sized variant that strikes a balance between network depth and training cost. Although ResNet-50 was released in 2015, it remains a noteworthy model in the history of image classification. In this project, we used a pre-trained ResNet-50 model to detect the weather in videos in real-time [11]. This model was trained on a dataset consisting of 65,000 images with six weather labels: sunny, cloudy, rainy, snowy, haze, and thunder. The model was trained for a total of 40 epochs, ultimately achieving an accuracy of 0.76 and a loss of 1.52 on the validation set. In our actual use, this model performed well, with generally accurate detection of the weather in the videos.

3 Moving Object Tracking

3.1 IOU Tracker

The IOU Tracker is an object tracking algorithm whose core idea is to determine whether objects in adjacent frames are the same by calculating the overlap (Intersection over Union, IOU) between them. In the code iouCommon.py, the IOUTracker class implements the IOU tracker. When receiving new detection results for a frame, the IOUTracker compares each detection result with existing trajectories. If the IOU value is greater than a preset threshold, the two results are considered to correspond to the same object, and the trajectory information of that object is updated. Otherwise, the detection result is treated as a new object, and a new trajectory is created for it.

3.2 Kalman filter

To more accurately predict the trajectory of the motion of objects, the system also introduces the Kalman filter algorithm. The Kalman filter is an algorithm used to estimate the state of linear systems. It can predict an object's position in the next frame based on its previous position and velocity, and correct the prediction result after receiving new observations (i.e., detection results). In the code iouCommon.py, the function create_kalma_filter creates a Kalman filter based on the KalmanFilter class from the OpenCV package [1]. This function initializes the state variables (object position and velocity), measurement variables (observed object position), transition matrix (describing the change of object state over time), measurement matrix (describing the relationship between observations and state variables), and error covariance matrix (describing the uncertainty of state estimation) of the Kalman filter. Additionally, the update_direction function estimates the object's motion direction (north or south) based on the velocity from the Kalman filter, which is used to count the traffic flow in the north and south directions. The update function in the IOUTracker class calls the create_kalman_filter function to create a Kalman filter for each new trajectory and utilizes the Kalman filter algorithm to predict the object's motion trajectory, thereby improving the accuracy and stability of tracking.

4 Lane line detection

4.1 ROI(region of interest)

Lane line detection is an important component of the traffic flow monitoring system, as it can help the system identify whether vehicles are deviating from their lanes or changing lanes illegally. In the code main.py, the detect_lane_lines function implements the lane line detection functionality. To improve efficiency and accuracy, this function first defines a region of interest (ROI) and performs lane line detection only on the image within the ROI. The definition of the ROI needs to be manually adjusted according to different video scenes. In this program, a polygon mask is used to define the ROI. The detect_lane_lines function first converts the image to grayscale and then applies Gaussian blur to reduce the impact of image noise. Next, it uses the Canny operator to extract edge information from the image and performs a bitwise AND operation between the edge information and the ROI mask to obtain the edge information within the ROI.

4.2 Hough transform

After obtaining the edge information within the ROI, the detect_lane_lines function uses the probabilistic Hough transform (HoughLinesP) to detect solid lines within the ROI, which represent the lane lines. The Hough transform is an algorithm used to detect straight lines in images. It can map points in the image space to curves in the parameter space and detect straight lines by counting the peak values at the intersection of curves. The detect_lane_lines function filters the straight lines based on preset thresholds, minimum line segment length, and maximum gap between line segments, and returns the detected lane lines. The is_vehicle_crossing_lane function is used to determine whether a vehicle is crossing the lane lines. It calculates the horizontal distance between the center point of the vehicle and the center point of the lane lines. If the distance is less than a preset threshold, the vehicle is considered to have crossed the lane lines.

5 GUI design and function integration

5.1 GUI Function Interface

GUI (Graphical User Interface) and function integration serve as the portal for the traffic flow monitoring system, where we integrate various functional modules to provide users with an intuitive operation interface. In the code main.py, the VideoGUI class is responsible for implementing the system's GUI, using the Tkinter package to create windows, buttons, labels, and other interface elements. The VideoGUI class includes multiple functional modules, such as video import, object detection and tracking, traffic flow statistics, traffic flow threshold setting, weather prediction, and system exit. Users can control the system's operation and view real-time detection results and statistical data through the buttons and input boxes provided by the VideoGUI class.

5.2 Real-time southbound/northbound traffic counts

The detect_video function in the VideoGUI class is responsible for implementing the system's core functions, including object detection and tracking, traffic flow statistics, and traffic flow threshold warning. The function first reads video frames and uses the IOUTracker class defined in the iouCommon.py program to track and count objects, displaying the southbound and northbound traffic flow on the GUI interface. To better manage traffic flow, we also provides a traffic flow threshold setting function. Users can set the southbound and northbound traffic flow thresholds through input boxes or choose to set thresholds automatically based on weather conditions.

5.3 Automatic traffic thresholds based on weather

To better manage the traffic flow, the system also provides a traffic flow threshold setting function. Users can set the southbound and northbound traffic flow thresholds through input boxes or choose to set thresholds automatically based on weather conditions. The system also integrates a weather prediction function that can predict weather conditions based on video frames and automatically adjust traffic flow thresholds accordingly. The weather prediction function uses the predict function defined in the code weather_prediction.py, which employs the PyTorch deep learning framework to load a pre-trained weather prediction model (a pre-trained ResNet-50 is used in our project, as mentioned in section 2.1) and make predictions on video frames. The get_weather_based_threshold function in the code main.py returns the corresponding traffic flow threshold based on weather conditions. This function uses a dictionary to store thresholds corresponding to different weather conditions, such as sunny, rainy, snowy, and cloudy. In the detect_video function, if the user selects the automatic threshold option, the system sets the southbound and northbound traffic flow thresholds based on the current weather conditions. If traffic flow exceeds the threshold, the system displays a warning message to alert the user about traffic congestion.

6 Result

The YOLOv5 series offers a total of five pre-trained models of varying sizes. We initially attempted to use the medium-sized YOLOv5m model, but the training performance was not so satisfying. The changes in various training metrics with respect to the number of epochs are shown in Figure 4.

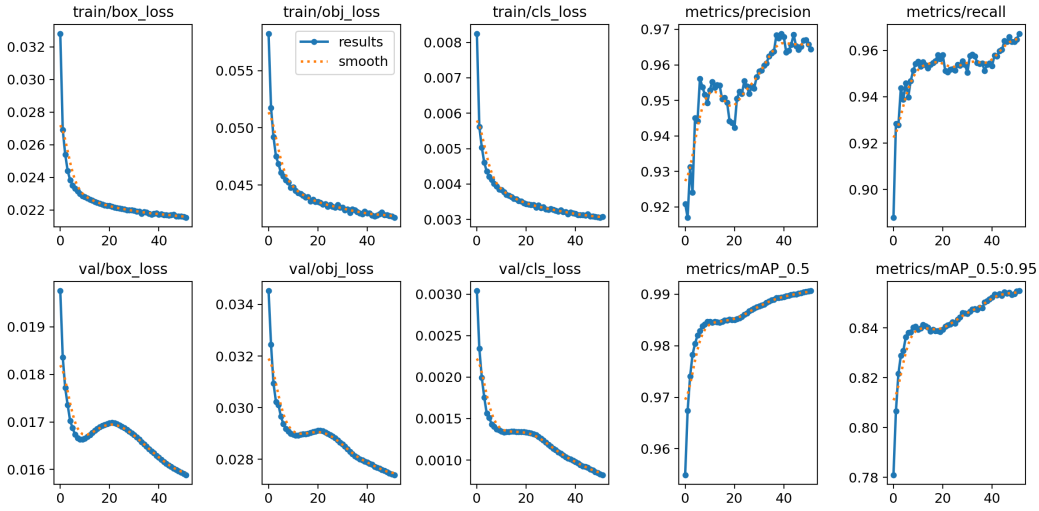

Figure 4: Training metrics with respect to the number of epochs of YOLOv5m

Therefore, we tried using the larger YOLOv5l model and trained it for 120 epochs. Its performance was significantly better than YOLOv5m, as shown in Figure 5. We considered experimenting with the even larger YOLOv5x model, but our devices were already struggling during the training of YOLOv5l, making it unlikely that we could successfully train YOLOv5x.

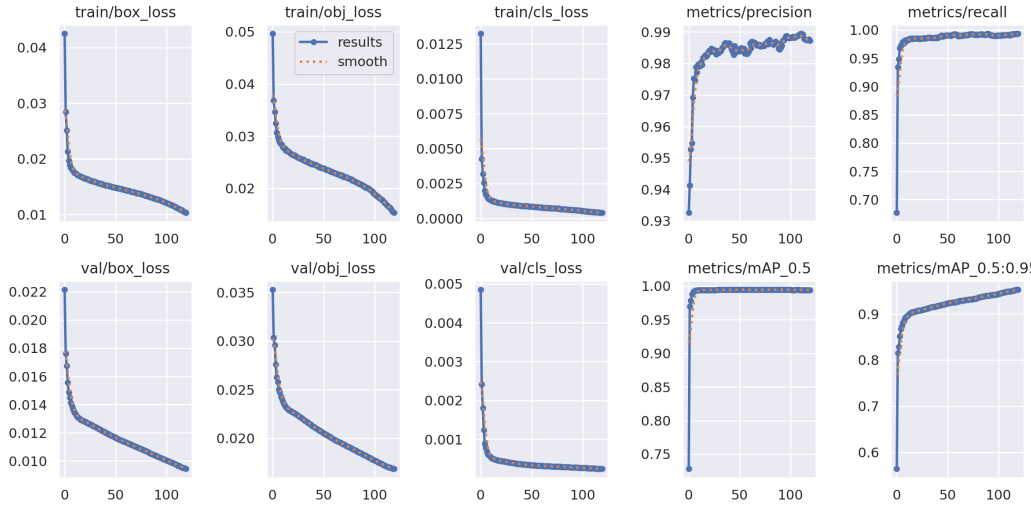

Figure 5: Training metrics with respect to the number of epochs of YOLOv5l

After training, the model can annotate each suspected vehicle in the frames of the video with a bounding box, as shown in Figure 6.

Figure 6: Illustration of the recognition results. The numbers on the bounding boxes represent the probability of identifying the object as the corresponding vehicle class.

We can set a threshold ourselves, and when counting vehicles, only objects with a predicted probability greater than the threshold are counted. The precision confidence curve obtained by training YOLOv5l on the UA-DETRAC dataset is shown in Figure 7.

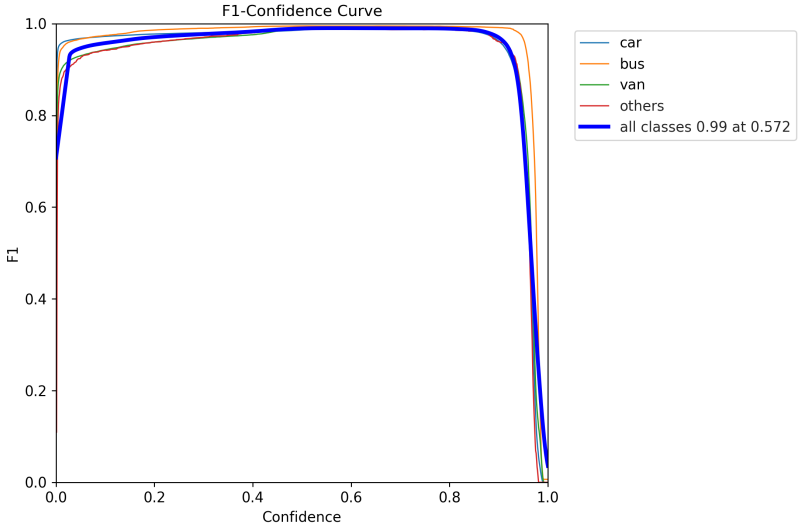

Figure 7: Training F1-Confidence curve of the model and dataset we choose ultimately.

To ensure that all vehicles are counted, we set the threshold to 0.5. Ultimately, our model successfully counted all the vehicles in the video frames.

7 Potential Improvements

There are several potential improvements that could be made to enhance the performance and applicability of our traffic monitoring system. Firstly, the vehicle recognition model was trained on a dataset with highly imbalanced vehicle type labels. The number of "others" (including vehicle types such as trucks and tankers) was less than 10,000, while the number of "cars" exceeded 400,000, as shown in figure 7. As a result, our trained vehicle detection model cannot accurately identify all vehicle types, limiting our ability to perform more granular traffic flow monitoring based

on vehicle type. Despite our efforts to find more usable datasets, we were unsuccessful, as the vast majority of vehicles on the road are indeed cars. Secondly, our model only detects the orientation of vehicle motion through Kalman filtering, without considering the vehicle's real-world velocity, which is an important factor in determining congestion. Although YOLOv8 has this capability, we understand that it may be slower than YOLOv5 when processing real-time video. Given that our current equipment is already approaching its performance limits when running YOLOv5 [7], we may only attempt to use YOLOv8 in the future when our hardware allows. Third, since we use the Hough transform method to detect straight lines for lane detection, this approach cannot handle curved lane lines. In the future, we could improve the model by addressing this limitation. Lastly, due to lack of time, we directly adopted an open source weather recognition model and did not use our own traffic dataset to train the weather model. If there is only a road in the video without obvious weather features (such as sky, rain, snow, thunderstorm), wrong weather conditions may be predicted. But this can be solved by simply training with our own traffic dataset annotated with weather labels.

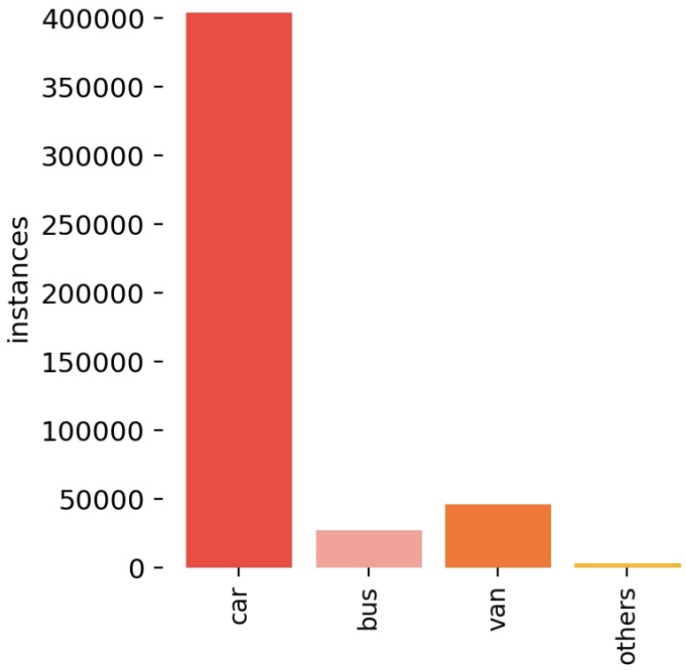

Figure 8: Highly imbalanced dataset

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Shenzhen Municipal Public Security Bureau. Vehicle management related business data for december 2023, 2024.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Glenn Jocher. YOLOv5 by Ultralytics, May 2020.

[7] E. G. Johnson. Yolov8 architecture vs yolov5. *Medium*, 2023.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[11] mengxianglong123. weather-recognition, 2022.

[12] Mario Pavlić, Heidrun Belzner, Gerhard Rigoll, and Slobodan Ilić. Image based fog detection in vehicles. In *2012 IEEE intelligent vehicles symposium*, pages 1132–1137. IEEE, 2012.

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

[14] Yuzhou Shi, Yuanxiang Li, Jiawei Liu, Xingang Liu, and Yi Lu Murphey. Weather recognition based on edge deterioration and convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2438–2443, 2018.

[15] Hai Wang, Yijie Yu, Yingfeng Cai, Xiaobo Chen, Long Chen, and Yicheng Li. Soft-weighted-average ensemble vehicle detection method based on single-stage and two-stage deep learning models. *IEEE Transactions on Intelligent Vehicles*, 6(1):100–109, 2020.

[16] R Wang, Y Xue, Z Li, Y Xue, and Z Li. Factors analysis on visible bands remote sensing images in the atmosphere [j]. *Journal of Gansu Normal Colleges*, 16(02):54–56, 2011.

[17] Zhangu Wang, Jun Zhan, Chunguang Duan, Xin Guan, Pingping Lu, and Kai Yang. A review of vehicle detection techniques for intelligent vehicles. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):3811–3831, 2022.

[18] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2020.

[19] Wikipedia contributors. Heihe–tengchong line — Wikipedia, the free encyclopedia, 2024. [Online; accessed 14-December-2024].

[20] L Xu, Z Jia, and X Qin. Detection and removal of snow from videos. *JOURNAL OF OPTO-ELECTRONICS LASER*, 18(4):478, 2007.

[21] Ashfak Yeafi. Road vehicle images dataset - bangladeshi road vehicle images with yolo v5 annotation, 2023.

[22] Xiaopeng Zhang, Hao Li, Yingyi Qi, Wee Kheng Leow, and Teck Khim Ng. Rain removal in video by combining temporal and chromatic properties. In *2006 IEEE international conference on multimedia and expo*, pages 461–464. IEEE, 2006.

[23] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9259–9266, 2019.