Counting Vehicles for Traffic Impact Analysis: A Vision-Based Approach

Amir Shahlaee (2022)

ABSTRACT

Traffic impact analysis stands as a crucial cog in the machinery of urban development, wherein the potential repercussions on traffic flow are meticulously scrutinized. Its significance cannot be overstated, as it plays a pivotal role in ensuring the continued safety and efficiency of transportation networks amidst the implementation of new projects. Traditionally, such analysis necessitates the parsing of vast amounts of data, particularly regarding traffic volume, gleaned from existing routes expected to bear the brunt of forthcoming developments. However, the conventional methodologies employed by the transportation sector for this purpose are laden with shortcomings—they are laborious, costly, and often disruptive to daily commuters. Herein lies our proposition: an innovative approach rooted in computer vision, leveraging neural networks to process real-time video feeds from traffic cameras. This promises a swifter, more cost-effective, and less intrusive means of garnering invaluable insights into traffic dynamics, encompassing volume and composition with unprecedented accuracy. In this scholarly exploration, we delve into the application of the rapid and resource-efficient YOLOv7 model to the realm of traffic data analysis, discerning its efficacy and potential for revolutionizing existing practices.

I. INTRODUCTION

In the realm of highway management, the ability to intelligently detect and count vehicles has become increasingly indispensable, given its wide array of practical applications spanning transportation safety and traffic management. The proliferation of traffic cameras across diverse road networks offers a potent resource for achieving this, obviating the need for costly infrastructural investments. By leveraging these cameras, planners gain the capacity to optimize city road networks efficiently, while concurrently establishing a robust database that serves as a springboard for enhancing roadway safety. However, amidst these promising prospects, significant challenges persist. Foremost among these hurdles is the inherent difficulty in accurately discerning vehicles of varying sizes, a challenge that hampers precise vehicle counts (Song et al., 2019). Additionally, factors such as the placement of cameras, background clutter, and fluctuations in vehicle position or orientation further compound the complexity of the task (Abdulrahim et al., 2016). Overcoming these obstacles is paramount to unlocking the full potential of intelligent vehicle detection and counting in highway management.

Traditionally, collecting traffic data has been a laborious endeavor, demanding costly hardware and extensive manual effort for maintenance and data preparation. In the past, methods such as loop detectors installed beneath road surfaces or surveillance cameras were employed, both requiring significant investment in terms of time and resources. For instance, as highlighted in a study by Shiranthika et al. (2019), pneumatic tubes have been utilized across urban roads in Australia for traffic monitoring. However, this approach is not without its drawbacks—it is relatively expensive, and the data retrieval process is time-consuming, taking up to two weeks. Moreover, the data collection process itself tends to be fairly random, providing limited insights into the frequency of road usage. Consequently, while these deployments offer some insight, they are not sufficiently comprehensive to meet the needs of the Australian Road Maritime Services for future road planning and administration. The high costs associated with the physical installation of such devices further exacerbate the limitations of these methods, rendering them impractical for widespread implementation across Australian road networks.

The primary objective of this study is to devise a vision-based approach capable of accurately counting vehicles while categorizing them into distinct types, such as cars, trucks, and buses. To explore this avenue, we construct a model leveraging the YOLO family of convolutional neural networks, utilizing publicly available datasets comprising images and videos sourced from traffic cameras. In the subsequent sections, we delineate the architecture of this model, elucidate the process of its development, and present its performance evaluation based on real-world traffic camera data encompassing diverse formats. Through this investigation, we aim to shed light on the efficacy and applicability of our proposed vision-based method for comprehensive vehicle counting and classification in varied traffic scenarios.

II. RELATED WORK

Shiranthika et al. (2019) introduced a real-time vehicle counting tool that tracks the centroids of moving objects, accurately calculates their speeds, and tallies them. Employing techniques such as background subtraction, morphological operations, and Gaussian filtering, this system is capable of issuing speed alerts when traffic congestion necessitates adjustments to highway speeds. Crucially, the real-time processing capability of the system has been achieved through meticulous optimization of OpenCV-based algorithms, requiring computing power comparable to that of a modern smartphone. Notably, the strategic positioning of cameras above roadways mitigates occlusion issues commonly encountered in vision-based traffic monitoring systems, rendering the technology resilient to adverse weather conditions prevalent on Australia's highways.

Addressing the challenge of accurately counting vehicles in low-resolution videos, Zhang et al. (2017) developed deep spatio-temporal neural networks tailored for sequential automobile counting from low-quality citycam footage. Their innovative FCN-rLSTM network, combining fully convolutional neural networks (FCN) with long short-term memory networks (LSTM) in a residual learning framework, effectively addresses the limitations posed by low resolution, slow frame rates, occlusion, and wide field of view. By leveraging the temporal dynamics captured by LSTM and pixel-level predictions enabled by FCN, their approach surpasses existing methods in vehicle counting accuracy.

In confronting challenges such as occlusion and background clutter, Wang and Liu (2022) proposed a strategy to enhance detection accuracy while maintaining real-time processing capabilities, particularly relevant for applications such as autonomous driving. Their approach incorporates AS-CBAM and HDC to maximize the receptive field and fine-tune characteristics, effectively mitigating background noise introduced by standard CBAM operations. Furthermore, the integration of DIOU-NMS and a tracking method based on Kalman filtering and Hungarian matching enhances the system's capacity to identify obscured objects, leading to heightened accuracy and suitability for autonomous driving scenarios.

Recent advancements in vehicle counting and volume estimation, as elucidated by Yang et al. (2021), underscore the importance of striking a balance between accuracy and speed in vision-based counting systems. By leveraging attention mechanisms and TSI density map estimation networks, their methodology achieves rapid and precise estimation of traffic volume and vehicle counts, even in scenarios with limited video data. This underscores the significance of their approach, which not only enhances counting accuracy and speed but also ensures robust performance under varied data conditions.

Song et al. (2019) proposed a methodology aimed at enhancing vehicle recognition by employing a novel segmentation technique to extract highway road surfaces and discern distant and proximal areas within images. Integrating these areas into the YOLOv3 network facilitates vehicle type and position determination. Additionally, leveraging the ORB algorithm for vehicle trajectory generation enables the identification of vehicle movement directions and total counts. Experimental findings demonstrate the efficacy of the segmentation strategy, particularly in enhancing the detection accuracy of smaller vehicle objects.

III. DATASETS

In our study, we harnessed a total of four datasets: one for training purposes and three for testing our model's efficacy. Each dataset is meticulously described in detail within subsections B and C, providing comprehensive insights into their composition and characteristics. Additionally, subsection A briefly outlines the challenges associated with procuring datasets tailored for vehicle counting problems, shedding light on the intricacies involved in sourcing appropriate data for research endeavors of this nature.

A. Challenges

As highlighted by Song et al. (2019), the availability of publicly accessible video and images from traffic cameras is severely limited due to concerns surrounding copyright, privacy, and security. While alternative datasets exist, such as those captured by on-vehicle cameras for autonomous vehicle training, their utility for developing systems to process data from traffic cameras is often compromised by differing shooting angles. The distinct perspectives captured by these datasets diverge significantly from the typical viewpoints of traffic cameras, thereby diminishing their suitability for training models geared towards processing real-world traffic camera data. This underscores the unique challenges inherent in accessing and utilizing appropriate datasets for research in the domain of traffic monitoring and vehicle counting.

B. Training Datasets

As elucidated in section IV, the models utilized in our study underwent pretraining on the MS COCO (Common Objects in Context) dataset, a resourceful repository provided by Microsoft for tasks in object detection, segmentation, and captioning. With a substantial collection of 300,000 images, over 200,000 of which are meticulously annotated, COCO encompasses roughly 1.5 million instances of objects across 80 predefined categories (Microsoft, n.d.). These categories span a diverse spectrum of items commonly encountered in real-world scenarios, encompassing everything from people and animals to various types of vehicles.

Importantly, COCO's predefined object categories align with the vehicle classes (car, bus, and truck) targeted by our model. Thus, models pretrained on COCO are inherently equipped to recognize these vehicle types. Through qualitative testing, we ascertained that the pretrained models exhibited impressive proficiency in vehicle recognition. Satisfied with these outcomes, we opted to retain the pretrained weights for each model and refrained from retraining them. This decision was reinforced by the robust performance observed during testing, affirming the efficacy of leveraging pretrained models in our vehicle counting framework.

C. Testing Datasets

In order to evaluate the performance of our model in real-world scenarios, mirroring the intended application of analyzing data from traffic cameras, we curated a selection of datasets sourced directly from highway cameras. This included two datasets comprising static images captured by highway cameras, providing a snapshot of typical traffic conditions. Additionally, we compiled a dataset consisting of videos captured by both traffic cameras and non-monitoring cameras, such as cell phones.

It's noteworthy that for the videos obtained from non-monitoring cameras, we took care to select footage captured from shooting angles akin to those of traffic cameras. This meticulous selection process ensured that the videos used for evaluation closely resembled the perspective and conditions encountered in actual traffic camera footage. By encompassing diverse sources of data, including both static images and videos, our evaluation aimed to comprehensively assess the model's performance across varied real-world scenarios encountered in traffic monitoring applications.

i. TRANCOS v3

The TRANCOS (Traffic and Congestions) dataset, compiled by Guerrero-Gómez-Olmedo et al. (2015), serves as a valuable resource for assessing vision-based vehicle counting solutions. Comprising 1,244 annotated images captured from surveillance cameras in Spain, this dataset is particularly notable for its portrayal of high-traffic conditions. Notably, the images in TRANCOS exhibit poor resolution and feature heavily overlapping vehicles, presenting a formidable challenge for vehicle counting algorithms.

Despite its complexities, we deliberately chose to incorporate the TRANCOS dataset into our testing regimen. Our rationale stemmed from the desire to ensure that our model could perform effectively across a spectrum of traffic conditions. Given that real-world traffic scenarios often involve congestion and high volumes of vehicles, it was imperative for our model to demonstrate robust performance under such challenging circumstances. A failure to accurately count vehicles in high-traffic situations would severely undermine the utility of any traffic counting model. By subjecting our model to the rigorous benchmarks set forth by the TRANCOS dataset, we sought to validate its efficacy and reliability in real-world traffic monitoring applications.

ii. Song et al. Highway Dataset

The Highway Dataset, curated by Song et al. (2019), offers a robust collection of 11,129 labeled images captured by highway surveillance cameras in China. Distinguished by their high resolution, these images predominantly depict sparse traffic conditions—a departure from the densely congested scenes characteristic of the TRANCOS dataset. This dataset presents an invaluable resource for evaluating vision-based vehicle counting solutions under less congested traffic scenarios.

We deliberately selected the Highway Dataset for testing purposes owing to its abundance of high-quality images and representation of sparse traffic conditions. It was imperative for our model to demonstrate proficiency in analyzing such scenarios effectively, alongside its ability to handle dense traffic conditions. By including the Highway Dataset in our evaluation, we aimed to comprehensively assess the versatility and robustness of our model across varying traffic densities. This diverse testing approach ensured that our model's performance was rigorously evaluated under a spectrum of real-world traffic conditions, thereby enhancing its reliability and applicability in practical traffic monitoring settings.

iii. Video for Object Detection YouTube playlist

The Video for Object Detection YouTube playlist, curated by Nakarin Lamangthong, offers a valuable compilation of 11 videos and a livestream, serving as informative resources for developing object detection solutions. Notably, the playlist features footage showcasing vehicles in various road scenes, with the livestream and five videos specifically focusing on vehicular activity. Collectively, these videos provide approximately one and a half hours of video data, offering diverse perspectives from six distinct locations.

We opted to utilize these videos for testing purposes with the objective of assessing our model's capability to analyze video feeds in real-time scenarios. Real-time analysis of video data is crucial for practical applications, particularly in traffic monitoring and surveillance. By subjecting our model to these videos, we aimed to evaluate its performance under dynamic, real-world conditions, ensuring its efficacy in processing streaming video data. This testing approach enabled us to validate the model's suitability for real-time vehicle counting applications, thereby enhancing its practical utility and reliability.

IV. NETWORKS

In our study, we opted to employ the YOLO (You Only Look Once) family of convolutional neural networks due to their renowned combination of high classification accuracy and rapid real-time classification capabilities, as noted by Bandyopadhyay (2022). Our vision entailed the development of a system capable of swiftly identifying and quantifying traffic patterns in real-time using live camera feeds. Consequently, we selected YOLO networks based on their suitability for this task.

Moreover, the YOLO models offer several practical advantages. They boast a user-friendly setup and usage process, making them accessible even to those with limited background knowledge in computer vision. Additionally, these models can deliver commendable performance on modest hardware setups, a feature that enhances their versatility and applicability across diverse computing environments. Furthermore, the abundance of online resources dedicated to YOLO networks facilitates their adoption and implementation, rendering them highly usable tools for transportation engineering groups seeking to integrate computer vision technologies into their traffic impact analyses.

We believed that leveraging the YOLO family's combination of usability and performance would facilitate the seamless integration of computer vision tools into traffic engineering practices, ultimately enhancing the efficiency and accuracy of traffic impact analyses conducted by transportation authorities.

A. General Features of YOLO Networks

The YOLO (You Only Look Once) family of convolutional neural networks comprises the original YOLO model, first introduced by Redmon et al. in 2016, alongside numerous derivative models that have emerged in the years since its inception. While these derivative YOLO models exhibit structural variations from the original, often tailored to enhance performance or cater to specific tasks, they all adhere to a conceptually similar architecture and object recognition algorithm.

At the core of the YOLO family's architecture is the "You Only Look Once" approach, which involves dividing the input image into a grid and predicting bounding boxes and class probabilities directly from this grid. This streamlined architecture enables YOLO models to achieve impressive speed and efficiency in object detection tasks, as the network processes the entire image in a single forward pass.

Despite structural differences among derivative models, such as adjustments in network depth, layer configurations, or the incorporation of additional features like skip connections, each model within the YOLO family maintains fidelity to the fundamental principles of the YOLO approach. This consistency in architecture and object recognition algorithm across the YOLO lineage facilitates seamless integration and experimentation with different model variants, empowering researchers and practitioners to select the most suitable YOLO model for their specific applications while retaining a foundational understanding of the underlying methodology.

i. Architecture

The original YOLO model, as presented by Redmon et al. (2016), is structured with inspiration drawn from the GoogleNet architecture. Figure 1 illustrates the architectural layout of YOLO, which encompasses 24 convolutional layers incorporating max pooling, along with two fully connected layers. This configuration enables the model to efficiently process input images and generate predictions for object detection tasks.

Derived YOLO models typically adhere to the architectural blueprint established by the original YOLO model. While variations exist to optimize performance or cater to specific objectives, the foundational framework remains consistent across the YOLO family. These derivative models may introduce modifications in layer configurations, network depth, or the incorporation of additional features, yet they maintain fidelity to the core principles underlying the YOLO approach.

Overall, the YOLO family's architectural coherence facilitates ease of understanding, experimentation, and implementation across different model variants, enabling researchers and practitioners to leverage the strengths of YOLO-based models while tailoring them to meet diverse application requirements.

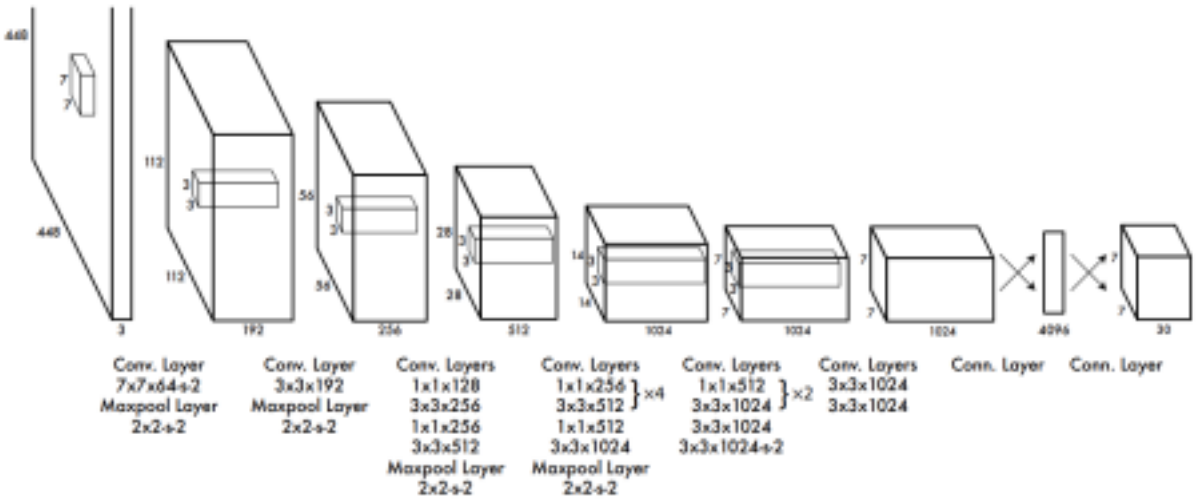

Figure 1. Original YOLO model architecture. The original YOLO model features 24 convolutional layers and two fully connected layers. Derivative YOLO models follow much the same architecture.

ii. Object recognition algorithm

Prior to the advent of YOLO, object recognition models commonly employed a two-stage approach to detect and classify objects in an image. These models typically utilized region proposal networks to identify candidate regions likely to contain objects, followed by classifiers to classify these regions. Although effective, this two-stage process necessitated multiple analyses of each image, rendering it too slow for real-time detection (Bandyopadhyay, 2022).

In contrast, YOLO revolutionized object detection with its singular approach, performing detection and classification as a unified task. This distinctive feature, from which YOLO derives its name, enhances both accuracy and speed compared to traditional two-stage detection models (Redmon et al., 2016). The algorithm employed by YOLO is conceptually straightforward: it first divides the input image into a grid of equally sized square cells. Each cell independently detects and localizes objects within it, generating predictions for object class, bounding box coordinates, and confidence score.

Crucially, YOLO's grid cells operate autonomously, without considering predictions from neighboring cells. Consequently, this approach often results in multiple duplicate or highly overlapping bounding boxes for the same object. To address this issue, YOLO employs a process known as non-max suppression in its final step. This process identifies the highest-probability classification for each object and suppresses substantially overlapping lower-probability classifications, ensuring that only one classification for each object remains.

By seamlessly integrating detection and classification into a single task and implementing efficient post-processing techniques like non-max suppression, YOLO achieves superior accuracy and speed in object detection, making it particularly well-suited for real-time applications in diverse domains, including traffic monitoring, surveillance, and autonomous vehicles.

B. Model 1: Pretrained YOLOv3

The initial model we employed for our study was a PyTorch implementation of the YOLOv3 model, which had been pretrained on the MS COCO dataset. YOLOv3, developed by Joseph Redmon and Ali Farhadi in 2018, is a prominent iteration within the YOLO family of convolutional neural networks. For our study, we obtained the pretrained YOLOv3 model from Erik Linder-Norén, an independent machine learning researcher, via GitHub.

To assess the model's performance, we conducted light qualitative testing using images sourced from the TRANCOS and Song et al. datasets, specifically focusing on its accuracy in vehicle detection tasks. However, our testing revealed that the model did not meet our expectations, often exhibiting shortcomings in identifying vehicles, particularly those located in the foreground of clear images. Dissatisfied with these results, we made the decision to explore newer YOLO models in the hopes of identifying a model that could deliver improved performance.

This decision to investigate newer YOLO models reflects our commitment to rigorously evaluating and optimizing our approach to achieve the desired accuracy and efficacy in vehicle detection. By exploring alternative models, we aimed to identify a solution that would better align with the objectives and requirements of our study.

C. Model 2: Pretrained YOLOv3 in Darknet Framework

The second model we explored was also a YOLOv3 variant pretrained on the MS COCO dataset, but it leveraged the Darknet framework—a neural network framework optimized for CUDA and GPU computation. We observed Darknet's widespread usage in various video detection applications of YOLOv3. Initially released in 2013 by Joseph Redmon, the creator of the original YOLO model, Darknet is designed for Unix systems. To accommodate our Windows environment, we obtained a compatible version from the GitHub repository of Alexey Bochkovskiy, a developer associated with newer YOLO models, starting from YOLOv4. From the same repository, we acquired the pretrained YOLOv3 model.

Conducting qualitative testing on this model using images from the TRANCOS and Song et al. datasets revealed a notable improvement in vehicle detection accuracy compared to our initial YOLOv3 model. Encouraged by these results, we extended our testing to include road traffic video clips sourced from YouTube. While the model exhibited superior performance, the testing also brought to light certain issues outlined in section V of our study. However, we encountered challenges in modifying the model within the Darknet framework to address these issues effectively.

Despite its improved accuracy, the difficulties in mitigating identified issues within the Darknet framework ultimately led us to discontinue further exploration of this model. This decision was motivated by the need to identify a solution that not only offered enhanced performance but also facilitated seamless modification and adaptation to meet the specific requirements of our study.

D. Model 3: Pretrained YOLOv7

The final model we explored was a YOLOv7 variant pretrained on the MS COCO dataset, akin to our previous investigations with YOLOv3 models. YOLOv7, touted as the fastest and most accurate YOLO model upon its release in 2022 by Wang et al., represented a significant advancement in object detection. We obtained a pretrained implementation of the YOLOv7 model from the GitHub repository of Wong Kin-Yiu, which is indicated in the YOLOv7 paper to be the official source for the published code.

Our evaluation revealed that the YOLOv7 model excelled in identifying vehicles, demonstrating superior speed and accuracy compared to the YOLOv3 models we had previously explored. Notably, we encountered no difficulties in modifying the YOLOv7 model, unlike our experience with the Darknet YOLOv3 model. Encouraged by its performance and ease of adaptation, we designated the YOLOv7 model as our "final" choice for the study.

YOLOv7 represents a significant departure from the original YOLO model, as it has undergone six

iterations of refinement. While a detailed description of its architecture is not explicitly provided in the YOLOv7 paper, an illustration of the standard YOLOv7 model's structure is included in an appendix to the paper, accessible via Wong Kin-Yiu's yolov7 GitHub repository. However, comprehensive descriptions of the architecture beyond this illustration have not been released. Nonetheless, the YOLOv7 model is characterized by convolutional layers, down- and upsampling layers, a cross-stage partial network, and efficient layer aggregation networks, as inferred from available resources and external research.

In subsequent subsections, we delve deeper into the architectural features and nuances of the YOLOv7 model, leveraging inference and external research to enhance our understanding of its design principles and functionality.

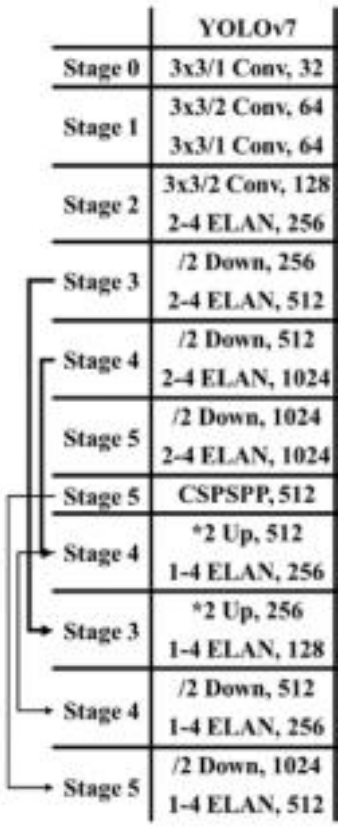

Figure 2. YOLOv7 model architecture. YOLOv7 features a greater variety of layer types than the original YOLO model and includes sub-networks.

i. Distinguishing features of YOLOv7

As a prominent iteration within the YOLO family, YOLOv7 distinguishes itself from earlier models through three notable architectural innovations, as outlined by Solawetz (2022).

Firstly, YOLOv7 incorporates efficient layer aggregation networks, representing a strategic layer aggregation strategy aimed at optimizing memory utilization in managing layers and backpropagation processes. This optimization contributes to enhancing the overall performance and efficiency of the model, ensuring streamlined processing without compromising accuracy.

Secondly, YOLOv7 introduces scaling optimization, which facilitates the preservation of optimal architecture when scaling the model up or down. This feature enables flexibility in adapting the model to different computational resources or application requirements while maintaining its efficacy and performance.

Thirdly, YOLOv7 integrates deep supervision by incorporating an auxiliary detection head near the middle of the network. During training, this auxiliary head operates in tandem with the prediction head located at the end of the network, enhancing the model's learning capacity and enabling more robust feature representation throughout the network's depth.

Collectively, these distinctive architectural features culminate in making YOLOv7 the fastest and most accurate YOLO model to date, as observed by Solawetz (2022). By prioritizing efficiency, scalability, and enhanced learning capabilities, YOLOv7 sets a new standard in object detection, offering unparalleled performance across a diverse range of applications and computational resources.

V. METHODOLOGY

In this section, we outline our testing procedures and methodologies for fine-tuning the output of our final YOLOv7 model to align with the objectives of our study. Throughout these processes, we leverage PyTorch and OpenCV utilities for Python to facilitate model development and analysis.

Our testing procedures involve evaluating the performance of the YOLOv7 model across various datasets and real-world scenarios, assessing its accuracy, speed, and robustness in vehicle detection tasks. We employ standard evaluation metrics, such as precision, recall, and mean average precision (mAP), to quantify the model's performance objectively.

Additionally, we explore methods to adjust and refine the model's output to optimize its relevance to our study goals. This may involve post-processing techniques such as non-maximum suppression, which helps eliminate duplicate or overlapping bounding boxes, or incorporating additional heuristics to improve the accuracy of object detection.

Throughout these processes, we prioritize the iterative refinement of our procedures based on

empirical observations and feedback from testing. We aim to ensure that the model's output aligns closely with the specific requirements and objectives outlined in our study, ultimately enhancing its utility and effectiveness in real-world applications.

By leveraging the capabilities of PyTorch and OpenCV, we streamline the development and testing processes, enabling efficient experimentation and refinement of our YOLOv7 model to achieve optimal performance in vehicle detection tasks.

A. Adding Procedures to the Model

As highlighted in section III/B, our decision to retain the pretrained weights of our model was based on the satisfactory performance it demonstrated in accurately identifying vehicles. Given this, we opted not to retrain the model from scratch, acknowledging the effectiveness of its existing pretrained weights.

However, despite the model's proficiency in vehicle identification, we recognized the need to implement additional procedures to fine-tune its output and address any potential issues with vehicle classification. This necessitated the integration of post-processing techniques and heuristic adjustments to refine the model's predictions and enhance its overall performance.

Specifically, we developed procedures aimed at adjusting the output of the model to mitigate common challenges such as duplicate or overlapping bounding boxes, inaccurate class assignments, and false positives. These procedures were designed to augment the model's capabilities and ensure more precise and reliable vehicle classification.

By incorporating these adjustment procedures into our workflow, we aimed to optimize the model's performance without necessitating extensive retraining. This approach allowed us to leverage the strengths of the pretrained model while addressing specific issues and fine-tuning its output to better align with the objectives of our study.

In summary, while we retained the pretrained weights of our model due to its satisfactory performance in vehicle identification, we supplemented this with additional procedures aimed at refining its output and enhancing its classification accuracy. This balanced approach enabled us to maximize the effectiveness of the model while minimizing the need for extensive retraining.

i. Suppressing undesired classifications.

We implemented a procedure to suppress classifications of all COCO object classes except for car, bus, and truck. Non-vehicle classifications are excluded from the model's classifications list, and their bounding boxes are not drawn in output images or video segments. This approach reduces visual clutter and streamlines calculations in subsequent procedures, focusing solely on vehicle identification.

ii. Suppressing extremely overlapping classifications.

One prevalent issue we encountered with all pretrained YOLO models was their limited ability to

accurately distinguish between vehicle classes. Despite their proficiency in identifying vehicles, they often struggled with correctly categorizing them. For instance, a car might be erroneously classified as a bus, or a truck initially identified as such might later be classified as a car in subsequent frames of a video.

This inconsistency in class distinction resulted in instances where the models produced overlapping classifications for a single vehicle, as illustrated in Figure 3. Although these classifications significantly overlap, they are not filtered by YOLO's non-max suppression component because they represent distinct classifications, such as one for a car and another for a truck.

This issue underscores the challenge of precise vehicle classification and highlights the need for further refinement in the model's classification capabilities to improve its accuracy and consistency in distinguishing between different vehicle types.

Figure 3. Extremely overlapping vehicle classifications. All the YOLO models occasionally produced overlapping, but different, classifications for single vehicles, indicated in this image by red-and-white arrows. For example, the car at the bottom left of the image is classified as both car and truck, and these classifications have almost wholly coincident bounding boxes. This image was produced by the Darknet YOLOv3 model.

To address the issue of inaccurate counting and visual clutter caused by overlapping classifications, we implemented a procedure to manually suppress these instances. Similar to non-max suppression, this procedure operates after the model has made all classifications on an image, and non-vehicle classifications have been suppressed as described previously.

The procedure scans the list of bounding box coordinates, identifies classifications with significant overlap, and removes the lower-confidence classifications until only one classification for each object remains. We define significant overlap as instances where both the top-left and bottom-right bounding box corners of two classifications are within a certain pixel distance of each other. This distance is manually defined based on the dimensions of the input image or video.

iii. Adding a counting procedure

Our model aims to output the number of vehicles identified in static images and the total count over the duration of video segments. For static images, we designed a simple counting procedure that tallies the total number of classified objects and the individual counts for each vehicle type (e.g., car, bus, truck), reporting these tallies both on the console and overlaid on the output image alongside the bounding boxes of each classification.

However, devising a counting procedure for video data posed a more intricate challenge, primarily due to the risk of overcounting. To address this concern, we implemented an approach where vehicles are counted only when they traverse a specific point in the video. This point is defined as an imaginary line positioned at 90% of the input video's y dimension (1/10th from the bottom of the frame), with a manually estimated buffer above and below the line. The buffer value is tailored for each video based on factors such as vehicle speed and distance from the camera.

On each frame of the video, the procedure checks if the bottom-right corner of any bounding box falls within the space of the imaginary line ± the buffer distance. If a bounding box intersects this space, the procedure increments the overall vehicle count and the count for the specific vehicle type. Similar to static images, these tallies are overlaid on each frame of the output video, allowing viewers to observe the count increasing as vehicles cross the imaginary line, which is also visually represented in the video. Refer to Figure 4 for an example output video frame.

Figure 4. Vehicles and the counting line. This example output video frame depicts the line used to count vehicles (green), the bounding boxes and classifications for each of the vehicles in the frame, and the running totals of vehicles identified in the video. In this clip, two vehicles have been identified so far, both of which have been cars. None of the vehicles in this frame have passed through the counting line yet.

Our intention was to integrate DeepSORT, an object tracking model, to enhance vehicle counting in videos. DeepSORT would enable us to identify vehicles individually and track them across frames. This approach would allow us to accurately determine if each active vehicle had crossed the predefined invisible line, mitigating the risk of overcounting.

Despite our efforts, we encountered several dependency and compatibility issues when attempting to integrate DeepSORT with our model. Unfortunately, we were unable to resolve these issues before the conclusion of our research period.

B. Testing

To evaluate our model, we randomly sampled images from the TRANCOS and Song et al. (2019) datasets and extracted short video clips from the Video for Object Detection playlist. Subsequently, we processed this data through our model and assessed its performance in terms of volume accuracy (comparing the number of vehicles counted to the actual number in the data) and class accuracy (accuracy of vehicle predictions).

VI. RESULTS

In this section, we present the outcomes of the tests conducted on our model. Due to time constraints, we couldn't devise a procedure for testing large datasets, resulting in a limited number of quantitative test results. Therefore, the majority of the results discussed herein are qualitative in nature.

A. Flaws of the Model

We've identified significant flaws in our model that have influenced the results discussed in the subsequent subsections. The primary flaw lies in our counting procedure for video data. Due to the absence of object tracking implementation, our model can't discern whether a particular vehicle has crossed the counting line; it merely checks if any bounding box's bottom-right corner is in proximity to the line. Consequently, this leads to undercounting of fast-moving vehicles (as their bounding boxes may pass over the line between frames) and overcounting of slow-moving vehicles (as their bounding boxes may remain near the line in consecutive frames). The magnitude of undercounting is exacerbated by a smaller buffer value, while overcounting worsens with a larger buffer value.

Additionally, our model exhibits poor categorization ability, resulting in incorrect counts for specific vehicle types. This limitation may stem from using pre-trained models without retraining them with the same type of data intended for analysis. Our model is designed to analyze images and footage from monitoring cameras, whereas the MS COCO dataset primarily comprises, if not entirely, images from non-monitoring cameras.

B. Volume Accuracy

In general, we found the model's volume accuracy (number of vehicles counted vs. actual number of vehicles in the data) to be satisfactory. The model performed best in identifying vehicles that were larger in the image frame and clearly defined. However, it struggled with smaller, overlapping, or blurry vehicles, as depicted in Figure 5, which showcases its analysis of images from the TRANCOS and Song et al. (2019) datasets.

While the model exhibited improved performance in identifying vehicles in video data—particularly when vehicles are closest to the camera and thus more discernible—it displayed limitations in accurately counting them, as discussed earlier. The model frequently missed fast-moving vehicles and occasionally counted slow-moving vehicles multiple times. This occasionally led to total counts that closely approximated the true vehicle count, albeit humorously. However, we anticipate inconsistent results for all video data, especially in scenarios such as footage of a highway where vehicles are moving at high speeds.

Figure 5. Illustrative examples of model undercounting. These output images from the TRANCOS (top) and Song et al. (2019) datasets illustrate that the model often fails to identify vehicles that are small, overlapping, or blurry. The counter is omitted from the TRANCOS image so vehicle classifications towards the bottom of the image are not obscured.

C. Class Accuracy

We found the model's class accuracy (accuracy of vehicle predictions) to be commendable. However, we observed challenges in distinguishing between cars and trucks when vehicles were distant, and between trucks and buses when vehicles were closer. Cars were more frequently misidentified as trucks than vice versa, and trucks were often misidentified as buses compared to the reverse scenario. We hypothesize that a car's grill may resemble a truck's appearance from a distance, and the rectangular containers of large trucks may appear similar to the bodies of buses when closer to the camera, particularly if the front of the truck is outside the camera's field of view.

Furthermore, as discussed earlier, the model's counting accuracy tends to be inconsistent for video data, leading to both undercounting and overcounting in various situations. This inconsistency also affects volume and class accuracy.

VII. CONCLUSIONS AND FUTURE WORK

After developing and testing our vision-based model for identifying vehicles and recording traffic volume using real traffic camera data, we maintain a strong belief in the potential of computer vision as a fast, cost-effective, and powerful tool for conducting traffic impact analyses. However, we acknowledge that achieving this goal requires a more robust approach than the one employed in this study.

For future endeavors, we aim to expand the algorithm to count other mobile objects, such as people. Distinguishing and counting each individual object once becomes crucial, especially when multiple surveillance cameras are involved. To address this, we explored literature on person re-identification (Re-ID) algorithms, which aim to retrieve a person of interest from numerous non-overlapping cameras. Despite challenges like alterations in lighting and varied viewpoints, recent advancements like the AGW algorithm by Ye et al. (2021) show promising effectiveness in Re-ID tasks.

Another potential application of this tool is in counting the number of passengers boarding buses or trains at each station and recognizing their destinations. This information could help identify the most in-demand origin-destination pairs, allowing for better planning of transit systems on those routes.

Overall, we believe that leveraging computer vision in transportation planning can greatly simplify various tasks, including improving safety policies by identifying crowded areas. As demonstrated by our study, while there are challenges to overcome, the potential benefits of such technology are substantial.

REFERENCES

Abdulrahim, K., & Salam, R. A. (2016). Traffic surveillance: A review of vision based vehicle detection, recognition and tracking. *International journal of applied engineering research*, 11(1), 713-726.

Bandyopadhyay, H. (2022). *YOLO: Real-time object detection explained*. V7 Labs. https://www.v7labs.com/blog/yolo-object-detection

Bochkovskiy, A. (2022). *Yolo v4, v3 and v2 for Windows and Linux.* GitHub repository. https://github.com/AlexeyAB/darknet

Guerrero-Gómez-Olmedo, R., Torre-Jiménez, B., López-Sastre, R., Maldonado-Bascón, S., & Onoro-Rubio, D. (2015, June). Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 423-431). Springer, Cham.

Kin-Yiu, W. (2022). *Of icial YOLOv7*. GitHub repository. https://github.com/WongKinYiu/yolov7

Linder-Norén, E. (2022). *PyTorch-YOLOv3*. GitHub repository. https://github.com/eriklindernoren/PyTorch-YOLOv3

Microsoft. (n.d.). *COCO – Common Objects in Context*. https://cocodataset.org/#home

Redmon, J. (2013). *Darknet: Open source neural networks in C*.

https://pjreddie.com/darknet/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*

preprint arXiv:1804.02767.

Shiranthika, C., Premaratne, P., Zheng, Z., & Halloran, B. (2019, August). Realtime Computer Vision-Based Accurate Vehicle Counting and Speed Estimation for Highways. In *International Conference on Intelligent Computing* (pp. 583-592). Springer, Cham.

Solawetz, J. (2022). *What is YOLOv7? A complete guide*. Roboflow. https://blog.roboflow.com/yolov7-breakdown/

Song, H., Liang, H., Li, H., Dai, Z., & Yun, X. (2019). Vision-based vehicle detection and counting system using deep learning in highway scenes. *European Transport Research Review*, 11(1), 1

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, K., & Liu, M. (2022). YOLOv3-MT: A YOLOv3 using multi-target tracking for vehicle visual detection. *Applied Intelligence*, 52(2), 2070-2091.

Yang, H., Zhang, Y., Zhang, Y., Meng, H., Li, S., & Dai, X. (2021). A Fast Vehicle Counting and Traffic Volume Estimation Method Based on Convolutional Neural Network. *IEEE Access, 9*, 150522-150531.

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., & Hoi, S. C. (2021). Deep learning for person re-identification: A survey and outlook. IEEE transactions on pattern analysis and machine intelligence, 44(6), 2872-2893.

Zhang, S., Wu, G., Costeira, J. P., & Moura, J. M. (2017). Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In Proceedings of the IEEE international conference on computer vision (pp. 3667-3676).