Preprocessing via Deep Learning for Enhancing Real-Time Performance of Object Detection

Yu Liu, Kyoung-Don Kang
State University of New York at Binghamton
{yliu456, kang}@binghamton.edu

***Abstract*—Deep learning models have significantly improved object detection essential for traffic monitoring. However, these models' increasing complexity results in higher latency and resource consumption, making real-time object detection challenging. To address this issue, we propose a lightweight deep learning model called Empty Road Detection (ERD). ERD efficiently identifies and removes empty traffic images that do not contain any object of interest, such as vehicles, via binary classification. By serving as a preprocessing unit, ERD filters out nonessential data, reducing computational complexity and latency. ERD is highly compatible and can work seamlessly with any third-party object detection model. In our evaluation, we found that ERD improves the frame processing rate of EfficientDet, SSD, and YOLOV5 by approximately 44%, 40%, and 10%, respectively, for a real-world traffic monitoring video.**

***Index Terms*—Real-Time Object Detection, Empty Road Detection, Traffic Surveillance**

I. Introduction

Traffic congestion and accidents incurs significant costs and even losses of lives. In New York City, 102 hours are lost in traffic, costing a driver over $1594 in 2021 [1]. Visual traffic surveillance is a fundamental infrastructure required to mitigate traffic congestion, reduce the travel time, and effectively respond to incidents. In traffic surveillance, real-time object detection is essential to analyze the traffic flow and detect incidents/anomalies, if any. Recently, CNN (Convolutional Neural Network) models, such as Region-based CNN (R-CNN) [2], Fast R-CNN [3], Faster R-CNN [4], EfficientDet [5], SSD (Single Shot MultiBox Detector) [6], and YOLO models [7], [8], have significantly enhanced the object detection performance, in terms of accuracy, precision, and recall.

Real-time object detection requires a high frame processing rate of at least 30 fps (frames per second). However, it can be difficult for deep learning models to achieve this speed due to their increasing complexity. To tackle the challenge, we propose a new lightweight CNN model that preprocesses each video frame to detect if there is any object of interest, such as a vehicle, motorcycle, or pedestrian. Our CNN model, called ERD (Empty Road Detection), filters empty traffic frames with no object of interest and forwards nonempty frames only to the object detection model to enhance the end-to-end frame processing rate. ERD detects empty frames with high accuracy of 0.96. In addition, *ERD is not designed for a particular object detection model but rather serves as a preprocessing unit that can be used in tandem with any third-party object detection model*. Unlike standalone lightweight CNN models such as MobileNetV3 [9], ShuffleNetV2 [10], SqueezeNet [11], EfficientNetV2 [12], and InceptionV3 [13], ERD has a much smaller model size and faster inference latency, as demonstrated in Table I. In this paper, we perform a case study to demonstrate the general applicability of ERD, where ERD works with three powerful object detection models, EfficientDet [5], SSD [6], and YOLOv5 [7], respectively.

Models	Parameters	GFLOPs	Latency@GPU	Latency@CPU
MobileNetV3-Small	2.5 M	0.27	8.4 ms	14.2 ms
ShuffleNetV2-X0	2.3 M	0.71	9.3 ms	20.3 ms
SqueezeNet1.1	1.2 M	1.74	5.5 ms	22.5 ms
EfficientNet-B0	5.3 M	1.88	12.2 ms	46.1 ms
EfficientNetV2-Small	21.5 M	13.4	25.1 ms	94.5 ms
InceptionV3	23.8 M	15.01	18.6 ms	80.9 ms
ERD (ours)	0.15 M	0.54	4.7 ms	10.2 ms

TABLE I: ERD vs. lightweight CNN models, processing with the images in size of 360×640.

A summary of our key contributions follows:

- We propose a new lightweight CNN model, ERD, to filter out empty traffic images to enhance the real-time performance of object detection in terms of fps. It supports high accuracy of 0.96. Also, its latency and resource consumption, i.e., the number of parameters and GFLOPS, are several times smaller than those of current lightweight models for complete object detection, as shown in Table I.
- We design an efficient pipeline consisted of our ERD model and an object detection model orthogonal to ERD, such as SSD, EfficientDet, or YOLOv5, to first remove empty frames and subsequently process unfiltered, nonempty traffic video frames, respectively.
- We have conducted extensive experiments for evaluation using a real-world traffic video [14]. By preprocessing frames and dropping frames estimated to be empty, ERD enhances the overall object detection fps. In our evaluation, ERD works together with EfficientDet, SSD, and YOLOV5, respectively. In this way, ERD enhances the fps of EfficientDet, SSD, and YOLOVv5 by approximately 44%, 40%, and 10% for the real-world traffic monitoring video, respectively. The ERD + YOLOv5 pipeline, which supports the highest fps among the tested combinations, supports over 30 fps per stream for up

to three concurrent video streams by efficiently utilizing resources. Furthermore, we analyze the performance impact of ERD for different proportions of empty frames in the traffic monitoring video. According to our results, the joint models, namely, ERD+EfficientDet, ERD+SSD, and ERD+YOLOv5, support higher fps than EfficientDet, SSD, and YOLOv5 when the percentage of empty road frames in a video exceeds about 7%, 17%, and 24%, respectively.

The rest of the paper is organized as follows. In Section II, our approach, ERD, is described. In Section III, the performance of ERD is compared to SSD, EfficientDet, and YOLOv5. Finally, we conclude the paper and discuss future work issues in Section IV.

II. Empty Road Detection via Deep Learning

In Figure 1, our proposed object detection pipeline consisted of ERD and an object detection model orthogonal to ERD, such as EfficientDet, SSD, or YOLOv5, is depicted. In Stage 1, ERD predicts whether the current video frame is empty or not. If ERD predicts that the frame is nonempty, we execute Stage 2 to detect and classify objects in the frame. Otherwise, we skip Stage 2.

In this paper, we design a lightweight CNN, ERD, which classifies a traffic monitoring image into empty or nonempty classes. We opt to design a CNN, since CNNs are very effective for computer vision. Before designing ERD, we have applied a traditional computer vision method, namely, background subtraction [15], but it provided low accuracy of 0.89 only. Also, we have analyzed the applicability of VGG-16 [16], ResNet-50 [17], and the lightweight models in Table I to detect empty traffic images. Although their accuracy is as high as ERD, they perform complete object detection unlike ERD, leading to significantly higher latency and resource consumption, as summarized in Table I.

In general, there is a tradeoff between prediction accuracy and latency. To strike a balance between the potentially conflicting requirements for high accuracy and low latency, we propose a new lightweight CNN model, ERD. We have explored a comprehensive set of CNN architectures with the varying depth and width, i.e., the number of the layers and neurons in each layer, and chosen the architecture with high accuracy and real-time performance. As illustrated in Figure 1, ERD consists of 13 layers that are grouped into 6 blocks: the first 5 blocks are convolutional, and the 6th block consists of the 3 fully-connected layers. Max/average pooling, being parameterless, is not considered a separate layer. Through the 5 convolution blocks, ERD squeezes the width and height of the feature maps, while increasing their depth: the input image is $360\times640\times3$ and the feature map produced by the last convolution layer is $22\times40\times32$, as shown in the figure. Each convolutional block consists of two convolutional layers. Thus, as depicted in Figure 1, ERD has 10 convolutional layers and 3 fully connected layers. In each convolution layer, ERD applies batch normalization and uses the rectified linear unit (ReLU) to speed up gradient descent, while mitigating possible gradient dispersion.

Figure 1 shows that the input image has a resolution of 360×640 with 3 channels (RGB). In ERD, the first two convolutional layers each use 8 kernels for 2D convolution. Each kernel in ERD is 3×3 in size with stride 1. The two $360\times640\times8$ feature maps produced by the convolution layers are concatenated and down-sampled into the $180\times320\times16$ feature map via 2×2 max pooling. Each of the first 4 convolution blocks consists of two convolution layers followed by max pooling. In the 5th convolution block, two convolutional layers are followed by 2×2 average pooling. By using max/average pooling, we compress the features maps to enable efficient inference of the road status (empty or not). Moreover, we aim to enhance the generalizability of our CNN model by considering features maps with different levels of variance, e.g., different degrees of illumination. When the average pooling in the 5th convolution block completes, the output feature map is flattened and input into the first fully connected (FC) layer with 32 neurons, as illustrated in Figure 1. ERD also has another FC32 layer, and finally an FC2 layer with two neurons. In this way, ERD converts the semantic features of the input image learned by the convolution blocks to the probability of the road being empty or not.

In total, ERD has 146K parameters only; it is smaller than the state-of-the-art CNN models by several orders of magnitude. For example, AlexNet [18], the first CNN model for image classification, which substantially outperformed classical low-level computer vision techniques not based on deep learning, has over 62 million parameters. The smallest YOLOv5 model [7] has more than 7 million parameters. Furthermore, ERD is significantly smaller and faster than the lightweight models for full object detection in Table I.

III. Evaluation

In this section, we evaluate performance in terms of fps, i.e., the number of frames processed per second by the object detection system. For our evaluation, we use a commodity machine with an Intel® Corei7-7820X CPU, 64 GB RAM, and a GeForce GTX 3080Ti GPU. Our operating system is Ubuntu 18.04.6 LTS. We have used Python 3.9, Pytorch 1.13, and OpenCV-python 4.6.0 [19] to implement and evaluate deep learning models.

A. Dataset

The dataset contains the images extracted from a video recorded on the road from the Bolshoy Moskvoretsky Bridge to Kremlin Embankment in Moscow, Russia. It is published on YouTube for computer vision research and development [14]. We extract the images of the whole video and label each image 0 or 1: empty or nonempty. Since only 21% of images are labeled 0, labels are imbalanced. To avoid overfitting and enhance generalizability, we augment the dataset by synthesizing every image labeled 0 by rotating, flipping, lighting, and blurring them. As a result, the fraction of empty images with no object of interest has increased to 35%, obtaining total

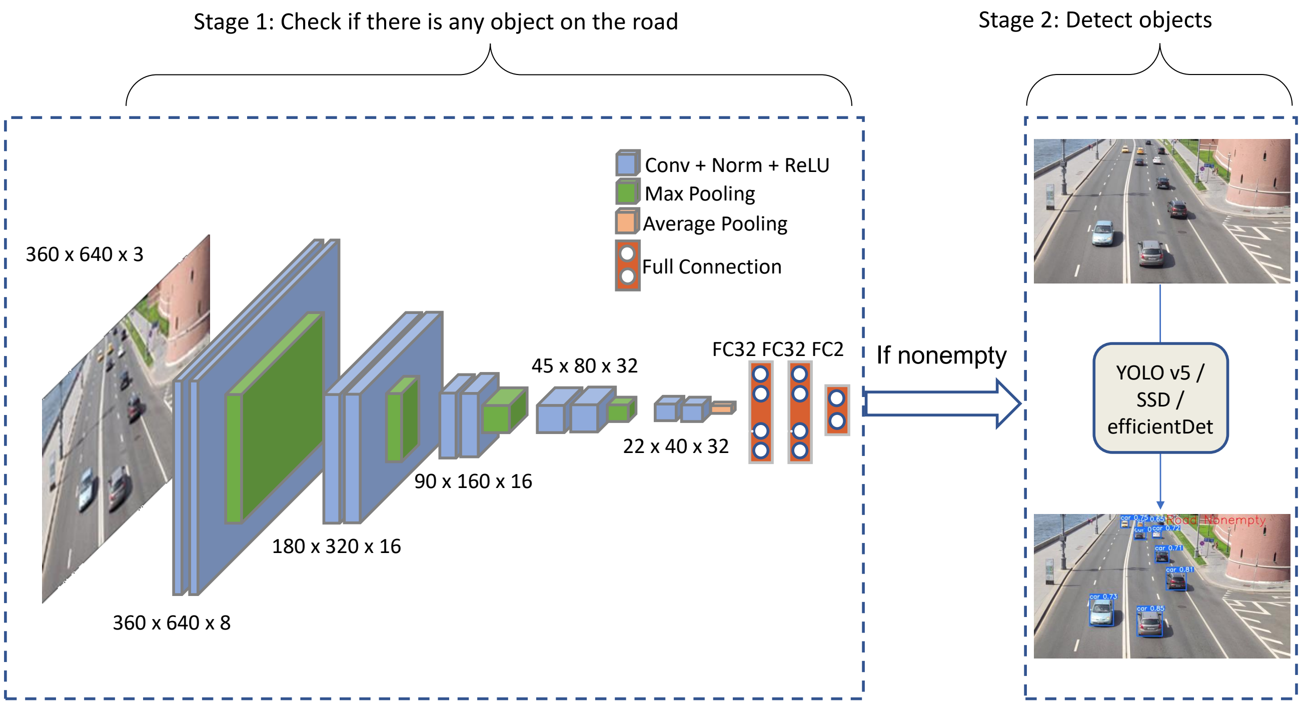
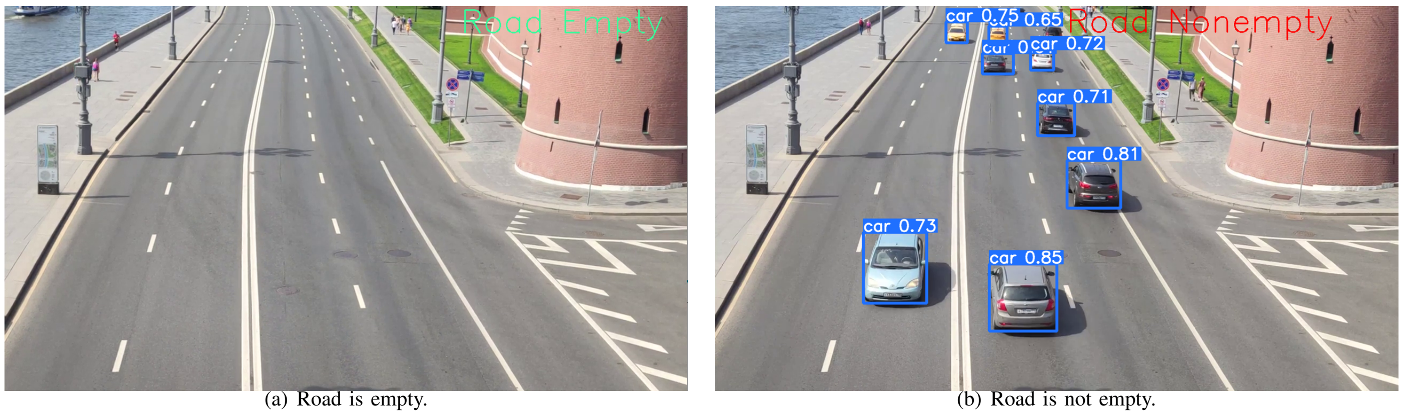


Fig. 1: Our object detection pipeline consists of the ERD model and an orthogonal object detection model.



(a) Road is empty.

(b) Road is not empty.

Fig. 2: An example of ERD + YOLOv5 results.

6438 images with 720×1280 resolution. We split the dataset into the training, validation, and testing set with the ratio of 0.6:0.2:0.2.

B. Object Detector

Our system is flexible in that it can work with any object detector to enhance the real-time performance of object detection by dropping empty frames reliably. In this paper, we use EfficientDet [5], SSD [6], and YOLOv5 [20], which are effective one-stage object detection models. We select them for their superior real-time inference speed and similar detection accuracy to the other state-of-the-art object detection models, such as Region-based CNN (R-CNN) [2], Fast R-CNN [3], Faster R-CNN [4], and the previous versions of YOLO [8]. Specifically, we have chosen EfficientDet-D0, SSD300, and YOLOv5s for our study, because they are smaller than their variants, still achieving high-quality object detection. This makes them desirable for real-time applications that require efficient and accurate object detection.

C. Evaluation Results

1) ERD accuracy: Figure 3 illustrates the confusion matrix of ERD. ERD misclassifies only 5 images that are empty as nonempty. Its accuracy is 0.96, recall 0.95, precision 0.97, F1-score 0.97.

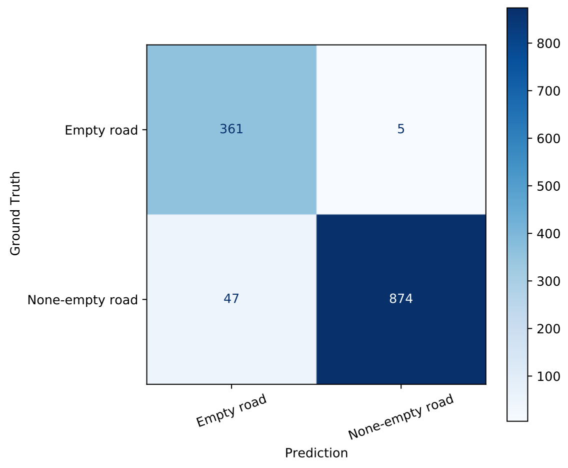


Fig. 3: Confusion Matrix of Empty Road Detection. Accuracy: 0.96, Recall: 0.95, Precision: 0.97, F1-score: 0.97.

2) Frame rate enhancements by ERD: In this paper, we run both ERD and object detection on the GPU to achieve significantly higher frame processing rates than the other configurations, which use the CPU for ERD and the GPU for object detection (or use the CPU for both), due to the massive parallelism provided by the GPU. In our experiments, we have observed that EfficientDet and YOLOv5 support the lowest and highest frame processing rate (fps), respectively (the results are omitted due to space limitations). Similarly, the ERD + YOLOv5 pipeline supports the highest fps, which is over 62 fps as shown in Table II. As summarized in Table II, ERD enhances the fps of the three tested pipelines using the different object detection models by approximately 10-44%. Notably, ERD achieves the highest fps improvement for EfficientDet, which is the slowest model. This is because removing empty frames via ERD allows the pipeline to run the slowest model less frequently, creating more impact on the overall performance.

Pipeline	ERD+YOLOv5	ERD+SSD	ERD+EfficientDet
Performance	62.4 (10.4%)	33.3 (39.9%)	13.7 (44.2%)

TABLE II: Frame processing rate improvements by ERD. In each cell, the first number is the total frame processing rate (fps) and the second number in the parenthesis is the enhancement of the fps achieved by ERD.

Due to the performance improvement brought by ERD, ERD + YOLOv5, which is the highest-performing scheme, can support more than 30 fps per stream for up to three concurrent streams as summarized in Table III. This is interesting because the aggregate frame rate of the three streams is over 91 fps, which is over 47% higher than the frame rate of the single stream of 62.4 fps in Table III. Therefore, ERD allows more efficient resource utilization to support real-time object detection for more concurrent streams.

Number of Streams	1 stream	2 streams	3 streams
Performance	62.4	53.1	30.6

TABLE III: Frame processing rate for multiple concurrent streams

3) Frame rates for different proportions of empty images: Intuitively, ERD can drop more frames if the video contains more empty frames, resulting in a higher fps for the ERD + object detection pipeline. However, ERD can impair the fps in an extreme case, where all frames are nonempty. In the rest of this section, we adjust the proportion of empty frames in the video described in Section III-A to perform a cost-benefit analysis of ERD. More specifically, we evaluate the frame rate for different proportions of empty frames in the traffic monitoring video, ranging from 0% to 100%, with a 10% increment. Thus, zero and all images are empty for 0% and 100%, respectively. As shown in Table IV, the fps monotonically increases as the proportion of empty frames in the video increases.

Pipeline	ERD+YOLOv5	ERD+SSD	ERD+EfficientDet
Performance	45.5−193.5	20.8−167.2	8.9−160.0

TABLE IV: Proportion of empty frames vs. fps. In each cell, the first and second number are the fps when 0% and 100% of the frames are empty.

In Figure 4, we present more detailed cost-benefit analysis results. Each dotted horizontal line shows the fps of a standalone object detection model without ERD, i.e., EfficientDet, SSD, or YOLOv5, which is not affected by the proportion of empty frames in the traffic video. It is worth noting that an ERD + object detection outperforms the corresponding object detection model without ERD when the percentage of empty frames is beyond a break-even threshold. Table V shows the break-even threshold for the tested pipelines. The slowest EfficientDet model and the fasted YOLOv5 model have the lowest and highest threshold, respectively.

These results suggest that ERD can be turned off during rush hour. Alternatively, offline profiling can be performed to identify the break-even threshold for an ERD + object detection configuration selected by the system administrator, e.g., the ERD + YOLOv5 pipeline, using an existing traffic monitoring video. At runtime, the percentage of empty frames can be continuously monitored to turn on ERD if the percentage exceeds the break-even threshold. A thorough investigation is reserved for future work.

Pipeline	ERD+YOLOv5	ERD+SSD	ERD+EfficientDet
Threshold	24.4%	17.3%	7.0%

TABLE V: Break-even thresholds of ERD. If the proportion of empty frames is higher than these thresholds, an ERD + object detection scheme supports higher frame processing rate than the corresponding standalone object detection model.

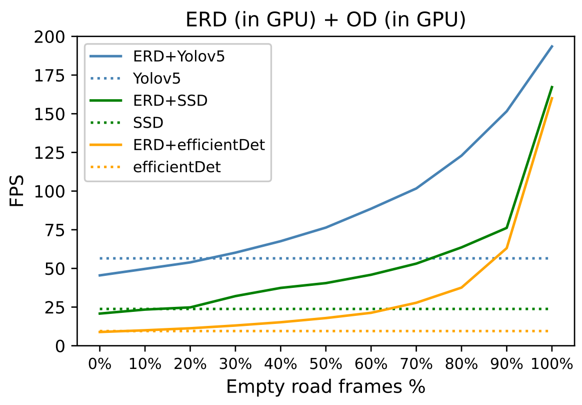


Fig. 4: Percentage of empty frames vs. fps

IV. Conclusions and Future Work

Traffic problems incur substantial societal costs. Real-time object detection is a core component in the traffic surveillance infrastructure. Advanced deep learning models developed recently have significantly improved object detection performance. Such models, however, are becoming deeper and more complex. As a result, their latency and resource consumption pose significant challenges for real-time object detection. In this paper, we propose a new lightweight CNN model, called empty road detection (ERD). ERD drops empty traffic images with no object of interest, such as vehicles or pedestrians, with high accuracy (0.96). By forwarding only nonempty images to the advanced object detection model, ERD improves the frame processing rate by 10-44% for the tested object detection models. The fastest combination, consisting of ERD and YOLOv5, supports over 30 fps/stream for three concurrent video streams. Additionally, ERD + EfficientDet, ERD + SSD, and ERD + YOLOv5 support higher frame processing rates than EfficientDet, SSD, and YOLOv5, when the proportion of the empty road frames in the video exceeds approximately 7%, 17%, and 24%, respectively. In the future, we will investigate more advanced approaches to further improve the real-time performance of traffic surveillance and deal with more general traffic scenarios.

Acknowledgement

This work was supported, in part, by the NSF award CNS-2007854. The authors thank anonymous reviewers who provided constructive feedback.

References

[1] NYC Ranks as the City With Worst Traffic Congestion in the U.S., Study Finds. https://www.nbcnewyork.com/news/local/nyc-ranks-as-the-city-with-worst-traffic-congestion-in-the-u-s-study-finds/3438472/, 2021. [Online] Accessed on Jan. 3, 2023.

[2] 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.

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[5] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[6] 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.

[7] YOLOv5. https://github.com/ultralytics/yolov5/wiki. [Online] Accessed on Nov. 27, 2022.

[8] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[10] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[11] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[12] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[13] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[14] FrançYouTube DZ Computer Vision. Traffic count, monitoring with computer vision. 4K, UHD, HD. https://www.youtube.com/watch?v=2kYpqSMqrzg, 2021. [Online] Accessed on Nov. 2, 2022.

[15] S Cheung Sen-Ching and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In *Visual Communications and Image Processing 2004*, volume 5308, pages 881–892. SPIE, 2004.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[17] 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 (CVPR)*, June 2016.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[19] opencv-python. https://pypi.org/project/opencv-python/. [Online] Accessed on Jan. 3rd, 2023.

[20] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.