Detecting Vehicles on the Edge: Knowledge Distillation to Improve Performance in Heterogeneous Road Traffic

Manoj Bharadhwaj
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
cs20s056@cse.iitm.ac.in

Gitakrishnan Ramadurai
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Civil Engineering
Indian Institute of Technology, Madras
gitakrishnan@civil.iitm.ac.in

Balaraman Ravindran
Robert Bosch Centre for Data Science and Artificial Intelligence
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
ravi@cse.iitm.ac.in

Abstract

The drastic growth in the number of vehicles in the last few decades has necessitated significantly better traffic management and planning. To manage the traffic efficiently, traffic volume is an essential parameter. Most methods solve the vehicle counting problem under the assumption of state-of-the-art computation power. With the recent growth in cost-effective Internet of Things (IoT) devices and edge computing, several machine learning models are being tailored for such devices. Solving the traffic count problem on these devices will enable us to create a real-time dashboard of network-wide live traffic analytics. This paper proposes a Detect-Track-Count (DTC) framework to count vehicles efficiently on edge devices. The proposed solution aims at improving the performance of tiny vehicle detection models using an ensemble knowledge distillation technique. Experimental results on multiple datasets show that the custom knowledge distillation setup helps generalize a tiny object detector better.

1. Introduction

With the recent advancements in Artificial Intelligence and Computer Vision, object detection techniques are gaining traction, especially in Intelligent Transportation Systems. Object detection for Intelligent Transportation Systems uses traffic cameras or other vision-based sensors. These techniques play a pivotal role in tackling challenging problems such as traffic density estimation, tracking traffic violations, lane changes, detecting speeds of individual vehicles, vehicle classification, and counting. With the recent developments in the field of machine learning for the Internet of Things (IoT), we can receive live analytics and predictions at a low cost using these devices [1, 19].

In this paper, we focus on the task of counting vehicles autonomously in an edge device using computer vision and deep learning. Estimating classified counts using cameras can be seen as the classic object counting task in computer vision [15]. Typically, object counting is the result of object detection and tracking. Therefore, we employ the Detect-Track-Count framework, consisting of three stages: vehicle detection, tracking, and counting. Since the framework is intended to be deployed on edge devices, the models deployed must be compact and computationally efficient. Given that tracking and counting are dependent

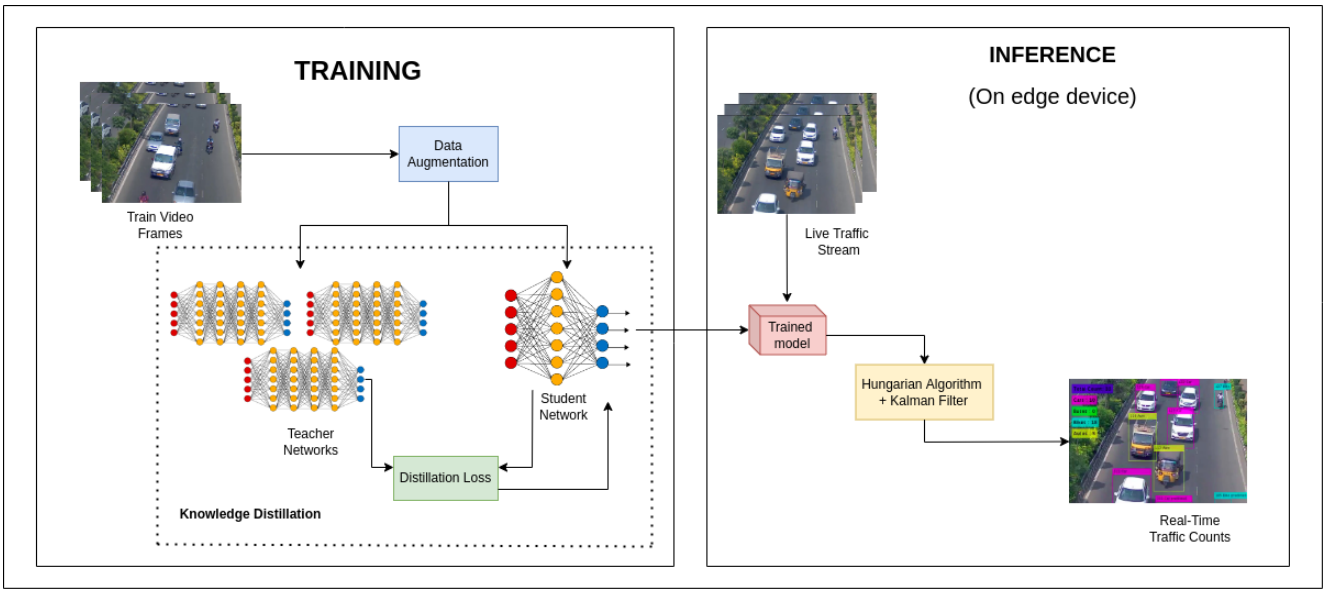


Figure 1. Proposed Architecture

on vehicle detection and have significantly lesser computations, we focus on making vehicle detection robust, fast, and lightweight.

Traditional vehicle detection techniques include Histogram of Oriented Gradients (HOG) [31], Scale Invariant Feature Transform (SIFT) [8], followed by Support Vector Machines(SVM) classification. These models are incredibly lightweight and can easily be fit on edge devices. However, these are not robust to several environmental factors (weather, time of the day), camera angle, and lighting. There emerged a need for a robust model that can handle any scenario in a real-time setup.

These defects migrated the interest of the research community towards deep learning models for object detection, such as Faster RCNN [30] (Region-based Convolutional Neural Networks), YOLO (You Only Look Once) [29], and SSD (Single Shot Detector) [23] for detecting the objects. These models are often very robust and handle dynamic inputs efficiently. However, these models are computationally demanding and need state-of-the-art computing systems to function.

The state-of-the-art vehicle detection for heterogeneous road traffic uses Faster RCNN, trained on Pascal VOC, augmented with a low-resolution dataset [26]. Faster RCNN is a computationally intensive model, and it is significantly slower than the frame rate of incoming video. As a result, deploying the model on an edge device with an average CPU and GPU is not possible. Faster RCNN can be replaced with YOLO to increase performance; however, even YOLO is computationally intensive on edge devices. Switching to even more lightweight models decreases the accuracy substantially.

To make these models lightweight, researchers have tried pruning techniques [25]. However, the increase in speed is not very significant, and if many layers are pruned, the performance degrades even further. Another approach to solving the problem is by using Knowledge Distillation [34]. This is an effective technique, inspired by the human's ability to quickly learn new complex concepts when given very small training sets [4, 14]. In model compression for deep learning, knowledge distillation has been widely used to transfer information from one network to another, i.e., a smaller student model is trained to mimic the performance of a large pre-trained teacher model or an ensemble of teacher models.

In this paper, anchoring on the computational capabilities of an Edge device, NVIDIA Jetson Nano, we propose a framework of vehicle counting that follows the detection-tracking-counting (DTC) model. We then use the 2021 AI City Challenge Track 1 [27] evaluation server to validate our counting accuracy and the computational efficiency of our model aligning with the goal of the AI City Challenge 2021's Track 1 to develop class-wise, motion-specific vehicle counting systems that can efficiently record vehicles in two categories, i.e., cars and trucks, in different traffic paths and scenes in real-time on a single-board computer.

Our contributions to the paper are the following:

- We improve the generalization of Tiny YOLO [29] by using an ensemble Knowledge Distillation technique [2] consisting of 3 different state-of-the-art object detection networks, thereby improving its detection ac-

curacy. Since Tiny YOLO is already lightweight, enhancing its detection accuracy will result in a better speed-accuracy tradeoff.

- In traffic scenes, when the vehicle density is high, motions usually accompany sudden changes in velocity and positions. We modify the tracker code and use an improved Kalman Filter model to circumvent these problems. We avoid using Deep Learning-based Trackers since they are computationally expensive.

- Finally, we compare our method on popular object detection datasets and use the AI City Challenge's evaluation server to validate our counts. Our experiments suggest that this implementation increases resource usage efficiency on an IoT device and effectively leverages a reasonably accurate DTC approach to traffic volume counting.

This is the first study, to the best of our knowledge, that uses ensemble knowledge distillation for object detection. We empirically validate the performance improvement produced when using this approach on the traffic domain by running extensive experiments on popular datasets [9, 22, 26].

2. Related Works

2.1. Object Detection

Object Detection is a task of computer vision to localize and classify objects. Object detection can be instrumental in vehicle identification [30], anomaly detection [28], lane identification [3], and pothole detection [10–12]. The research community has shifted its attention to deep learning to tackle the object detection problem in the past decade. Research results have indisputably demonstrated the effectiveness of deep neural networks (DNN) for accurately recognizing image objects [5]. However, the improvement in accuracy has reduced computation speed even with high-end machines [21]. Thus, we need model compression techniques to scale these models on edge devices [6, 20, 32, 33].

2.2. Knowledge Distillation

Knowledge distillation refers to the process of transferring the knowledge from a large, unwieldy model or set of models to a single smaller model that can be practically deployed under real-world constraints [16]. Knowledge Distillation is a popular technique for image classification [7].

This method can also be used as a model compression technique for object detection. However, in the case of object detection, multiple models differ significantly. For example, YOLO is a single-shot object detector, whereas Faster RCNN is a two-stage network. Therefore, distilling the layers of these networks together becomes an arduous task. Researchers use similar teacher and student models for the knowledge distillation process [13, 24, 37]. However, restricting to similar models limits the scope of performance improvement.

3. Proposed Vehicle Counting Framework

As shown in Fig. 1, the full pipeline can be divided into two stages:

- Training Phase
- Inference Phase

3.1. Training the Student Network

We train the Tiny YOLO architecture (student network) with an ensemble of multiple teachers, such as Faster RCNN, SSD, and YOLO. The student network is first initialized with Imagenet weights. During the entire training phase, the weights of the teacher networks are frozen. Then, for each batch, the teachers' predicted outputs are reshaped to a feature map of size 26×26, and the mean feature map of all the teachers is computed. While training the student, we use this mean feature map as the reference feature map for the student network along with a non-local module [38], which is an effective method to improve the performance of neural networks by capturing the global relation information [35]. Here, we apply the non-local module to capture the relation between pixels in an image. The loss function of the student network ($\mathcal{L}_s(z)$) is given by the following expression:

$$\mathcal{L}_s = \mathcal{L}_{sc} + \mathcal{L}_{so} + \mathcal{L}_{scl} + \mathcal{L}_{sYOLO} + \mathcal{L}_2(t, s) + \mathcal{L}_2(r^t, r^s)$$

- $\mathcal{L}_{sc}$ is the coordinate loss of the Tiny YOLO network. This loss checks if the object is covered entirely by the bounding box.

- $\mathcal{L}_{so}$ is the objectness loss of the Tiny YOLO network. This loss accounts for the box-object Intersection of Union (IoU) prediction.

- $\mathcal{L}_{scl}$ is the classification loss of the Tiny YOLO network. This is '1' for the correct classes and '0' for all the other classes for the object in a box.

- $\mathcal{L}_{sYOLO}$ is the special YOLO loss, which captures the deviation of the anchor box generated from its original anchor shape and location.

- $\mathcal{L}_2(t, s)$ is the $\mathcal{L}_2$ norm of the mean feature map of the teachers and the feature map of the student network.

- $\mathcal{L}_2(r^t, r^s)$ is the $\mathcal{L}_2$ norm of the obtained relational information of the teacher (mean relational information of all the teachers) and student network.

  After the student model is fully trained, it is further optimized using the Tensor RT module [17] to make the model more lightweight.

3.2. Estimating the Vehicle Counts

The trained student model is loaded into the edge device. We then use the standard Simple Online Real-time Tracker (SORT) [36] to estimate the vehicle's trajectories. The Kalman filter is a crucial component in SORT. Our state contains 8 variables; $(u, v, a, h, u', v', a', h')$ where (u, v) are centres of the bounding boxes, a is the aspect ratio and h, the height of the image. The other variables are the respective velocities of the variables.

We create a "Track" for each detection with all the necessary state information. We also have a parameter to track and delete tracks that had their last successful detection long back, as those objects would have left the scene. Also, there is a minimum number of detections threshold for the first few frames to eliminate duplicate tracks. Then, we use the Hungarian Algorithm [18] to assign IDs for the various tracks. These IDs are used to update the classified count of the vehicles.

4. Implementation Details

We ran our experiments on 3 standard object detection datasets, namely: Pascal VOC, IITM-Hetra, and COCO. No data augmentation was performed for Pascal VOC and COCO datasets. IITM-Hetra was augmented with Pascal VOC 2007 to improve the performance as the number of samples in IITM-Hetra was insufficient, even for the teacher networks. The vehicle detection training was performed in a GPU Cluster, having several GPU Machines like NVIDIA Titan X, NVIDIA 1080Ti, and NVIDIA DGX with GPU RAM sizes 12Gi, 11Gi, and 32Gi, respectively. We used the pre-trained teacher models (YOLOv3, Faster RCNN, and SSD) with the input size fixed to 416×416 and initialized the student network with Imagenet weights. The student model was trained for 100 epochs, with a batch size of 32. Adam was applied as the optimizer, and we decayed the initial learning rate from $3.5e^{-4}$ to $7.7e^{-7}$ using a cosine annealing scheduler and the decay parameter was set to e^{-1}. The inference was run on an edge device, **NVIDIA Jetson Nano**, which is a 128-core Maxwell, Quad-core ARM $A57$ @ 1.43 GHz processor with a 4Gi RAM.

5. Experiments and Results

For all the experiments, we use the Mean Average Precision (MAP) score and the average recall (AR) score as the metrics to compare.

5.1. Vehicle Detection

We experimented with multiple knowledge distillation setups to train the student network and compared the results with the vanilla Tiny YOLO architecture with no distillation. The baseline model was initialized with Imagenet weights and is trained on the various datasets.

1. Using YOLOv3 as the teacher network and applying distillation on feature map layers.

2. Using two teacher networks - YOLOv3 and YOLOv2, for the multiple layer setup. We chose the best-performing teacher for a batch.

3. Using 3 teachers (Faster RCNN, YOLOv3, and SSD), averaging the feature maps of the teachers and using them to train the student network.

The teacher models are very accurate in all the datasets and their mean mAP score is 0.74, while the mean AR score is 0.87.

Dataset	Model	mAP	AR
COCO	Tiny YOLO (Baseline) (No Distillation)	0.237	0.35
COCO	Tiny YOLO (Distilling multiple layers)	**0.241**	0.35
Pascal VOC 2012	Tiny YOLO (Baseline) (No Distillation)	0.31	0.42
Pascal VOC 2012	Tiny YOLO (Distilling multiple layers)	**0.33**	**0.44**
IITM-Hetra	Tiny YOLO (Baseline) (No Distillation)	0.32	0.41
IITM-Hetra	Tiny YOLO (Distilling multiple layers)	0.32	**0.412**

Table 1. Distilling multiple layers of a single teacher

Dataset	Model	mAP	AR
COCO	Tiny YOLO (Baseline) (No Distillation)	0.237	0.35
COCO	Tiny YOLO (2 Teachers, multiple layers)	**0.254**	0.356
Pascal VOC 2012	Tiny YOLO (Baseline) (No Distillation)	0.31	0.42
Pascal VOC 2012	Tiny YOLO (2 Teachers, multiple layers)	**0.35**	**0.45**
IITM-Hetra	Tiny YOLO (Baseline) (No Distillation)	**0.32**	0.41
IITM-Hetra	Tiny YOLO (2 Teachers, multiple layers)	0.31	**0.42**

Table 2. Distilling multiple layers of 2 Teachers

Dataset	Model	mAP	AR
COCO	Tiny YOLO (Baseline) (No Distillation)	0.237	0.35
COCO	Tiny YOLO (Multiple teachers)	**0.28**	**0.41**
Pascal VOC 2012	Tiny YOLO (Baseline) (No Distillation)	0.31	0.42
Pascal VOC 2012	Tiny YOLO (Multiple teachers)	**0.41**	**0.53**
IITM-Hetra	Tiny YOLO (Baseline) (No Distillation)	0.32	0.41
IITM-Hetra	Tiny YOLO (Multiple teachers)	**0.39**	**0.58**

Table 3. Distilling multiple Teachers

In Tables 1 and 2, we notice a slight improvement in the mAP scores and AR scores of the distilled models. These improvements, however, are not very significant. In Table 3, the increase in performance is evident. This could be because of incorporating multiple different teacher networks for knowledge distillation.

5.1.1 Visualizing the detection of the student network

We have visualized the predictions of the student model on various datasets and compared them with the baselines.

(a) Tiny YOLO Baseline (No distillation)

(b) Our model (Distilled from multiple teachers)

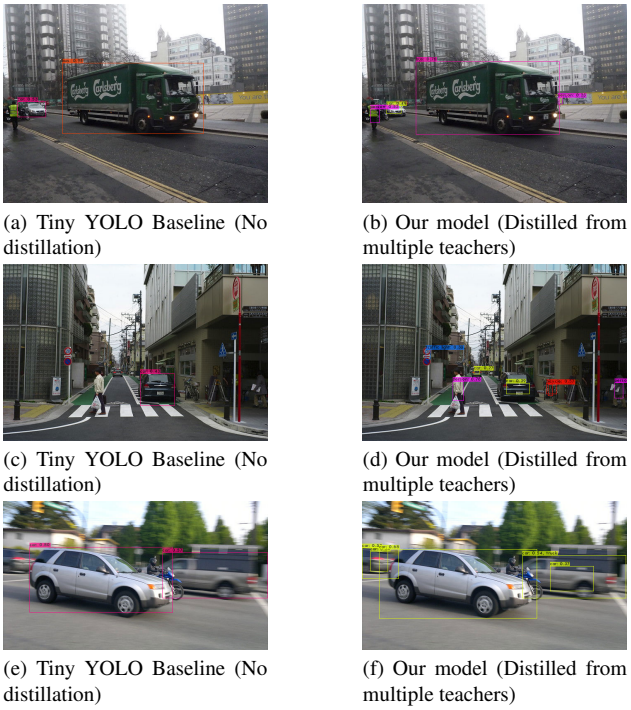
(c) Tiny YOLO Baseline (No distillation)

(d) Our model (Distilled from multiple teachers)

(e) Tiny YOLO Baseline (No distillation)

(f) Our model (Distilled from multiple teachers)

Figure 2. Vehicle Detection on Pascal VOC Dataset



(a) Tiny YOLO Baseline (No distillation)

(b) Our model (Distilled from multiple teachers)

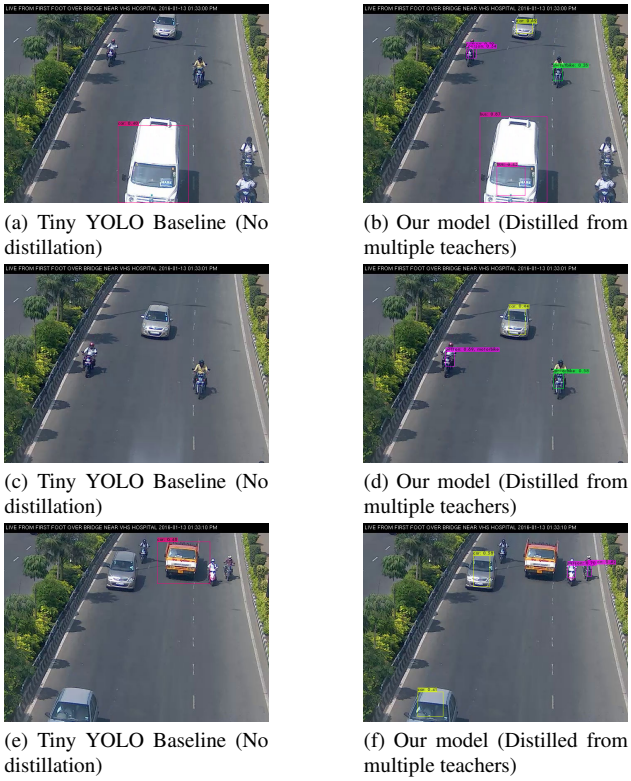
(c) Tiny YOLO Baseline (No distillation)

(d) Our model (Distilled from multiple teachers)

(e) Tiny YOLO Baseline (No distillation)

(f) Our model (Distilled from multiple teachers)

Figure 3. Vehicle Detection on COCO Dataset



(a) Tiny YOLO Baseline (No distillation)

(b) Our model (Distilled from multiple teachers)

(c) Tiny YOLO Baseline (No distillation)

(d) Our model (Distilled from multiple teachers)

(e) Tiny YOLO Baseline (No distillation)

(f) Our model (Distilled from multiple teachers)

Figure 4. Vehicle Detection on IITM-Hetra Dataset

Our proposed approach performs better than the vanilla Tiny YOLO network in many instances. The baseline Tiny YOLO model fails to detect efficiently in the complex examples like Figure 2e, where the baseline predicted a truck and car. As seen in Figure 2b, our proposed model is able to identify a lot more objects in the frame. However, there are overlapping detections, and these could be post-processed. There are instances where the vanilla Tiny YOLO architecture failed to return a single prediction (Ex: Figure 4c) in the IITM-Hetra dataset.

5.2. Vehicle Counting

The student model trained on IITM-Hetra and the YOLOv3 model trained on COCO are evaluated on the AI City Challenge 2021's Track 1 server to get the count accuracy. [27]

The metric used is the $S1$ score, which is defined as $S1 = 0.3 * S1_{Efficiency} + 0.7 * S1_{Effectiveness}$, where $S_{Efficiency}$ is calculated based on the execution time and is adjusted by the Base Factor which is dependent on the running system's CPU and GPU computation.

It assesses the solution's ability to execute online within its computing environment and resources. On the other hand, $S_{Effectiveness}$ is computed for vehicle counts as a weighted average of normalized weighted root mean square error scores (nwRMSE) across all videos, movements, and

vehicle classes of the test sets.

Model	$S1_{Effectiveness}$	$S1_{Efficiency}$	S1
YOLOv3 (Teacher)	0.9425	0.9911	0.9571
Tiny YOLO (Distilled) (Student)	0.5840	0.9995	0.7085

Table 4. Counting Accuracy

6. Conclusion

In this paper, we have presented a low-compute framework to estimate the traffic counts. Our solution consists of a fast object detector (Tiny YOLO) trained using Knowledge Distillation from 3 teacher networks (Faster RCNN, YOLOv3, and SSD), Tensor-RT Optimization of the learned model, and estimating the count of the traffic using the SORT Algorithm in real-time on edge devices. Both the Effectiveness and Efficiency of our solution are experimentally illustrated using multiple data sources. Based on the results, there is a significant potential for generalizing tiny models and improving performance.

References

[1] Rasheed Ahmad and Izzat Alsmadi. Machine learning approaches to iot security: A systematic literature review. *Internet of Things*, 14:100365, 2021. 1

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning, 2020. 2

[3] Abdulhakam AM Assidiq, Othman O Khalifa, Md Rafiqul Islam, and Sheroz Khan. Real time lane detection for autonomous vehicles. In *2008 International Conference on Computer and Communication Engineering*, pages 82–88. IEEE, 2008. 3

[4] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. 2

[5] Karanbir Singh Chahal and Kuntal Dey. A survey of modern object detection literature using deep learning. *CoRR*, abs/1808.07256, 2018. 3

[6] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[7] Wei-Chun Chen, Chia-Che Chang, Chien-Yu Lu, and Che-Rung Lee. Knowledge distillation with feature maps for image classification, 2018. 3

[8] Jae-Young Choi, Kyung-Sang Sung, and Young-Kyu Yang. Multiple vehicles detection and tracking based on scale-invariant feature transform. In *2007 IEEE Intelligent Transportation Systems Conference*, pages 528–533, 2007. 2

[9] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, jun 2010. 3

[10] Rui Fan, Xiao Ai, and Naim Dahnoun. Road surface 3d reconstruction based on dense subpixel disparity map estimation. *IEEE Transactions on Image Processing*, 27(6):3025–3035, 2018. 3

[11] Rui Fan and Ming Liu. Road damage detection based on unsupervised disparity map segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4906–4911, 2020. 3

[12] Rui Fan, Umar Ozgunalp, Brett Hosking, Ming Liu, and Ioannis Pitas. Pothole detection based on disparity transformation and road surface modeling. *IEEE Transactions on Image Processing*, 29:897–908, 2020. 3

[13] Heitor Felix, Walber M. Rodrigues, David Macêdo, Francisco Simões, Adriano L. I. Oliveira, Veronica Teichrieb, and Cleber Zanchettin. Squeezed deep 6dof object detection using knowledge distillation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. ISSN: 2161-4407. 3

[14] Steven Gutstein, Olac Fuentes, and Eric Freudenthal. Knowledge transfer in deep convolutional neural nets. *International Journal on Artificial Intelligence Tools*, 17(03):555–567, 2008. 2

[15] Synh Viet-Uyen Ha, Nhat Minh Chung, Tien-Cuong Nguyen, and Hung Ngoc Phan. Tiny-pirate: A tiny model with parallelized intelligence for real-time analysis as a traffic counter. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4119–4128, June 2021. 1

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 3

[17] Siddharth Sharma Josh Park, Sirisha Rella and Houman Abbasian. Speeding up deep learning inference using nvidia tensorrt. 3

[18] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 4

[19] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. Internet of things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data*, 6, 12 2019. 1

[20] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection, 2019. 3

[21] Lin Li, Shengbing Zhang, and Juan Wu. Efficient object detection framework and hardware architecture for remote sensing images. *Remote Sensing*, 11(20), 2019. 3

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 3

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. 2

[24] Rakesh Mehta and Cemalettin Ozturk. Object detection at 200 frames per second. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 659–675. Springer International Publishing. 3

[25] Deepak Mittal, Shweta Bhardwaj, Mitesh M. Khapra, and Balaraman Ravindran. Studying the plasticity in deep convolutional neural networks using random pruning, 2018. 2

[26] Deepak Mittal, Avinash Reddy, Gitakrishnan Ramadurai, Kaushik Mitra, and Balaraman Ravindran. Training a deep learning architecture for vehicle detection using limited heterogeneous traffic data. In *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, pages 589–294, 2018. 2, 3

[27] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021. 2, 5

[28] Khac-Tuan Nguyen, Dat-Thanh Dinh, Minh N. Do, and Minh-Triet Tran. *Anomaly Detection in Traffic Surveillance Videos with GAN-Based Future Frame Prediction*, page 457–463. Association for Computing Machinery, New York, NY, USA, 2020. 3

[29] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 2

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 06 2015. 2, 3

[31] Paul E. Rybski, Daniel Huber, Daniel D. Morris, and Regis Hoffman. Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In *2010 IEEE Intelligent Vehicles Symposium*, pages 921–928, 2010. 2

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[33] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, June 2021. 3

[34] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 2

[35] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[36] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017. 4

[37] Linfeng Zhang and Kaisheng Ma. Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. 3

[38] Linfeng Zhang and Kaisheng Ma. Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In *International Conference on Learning Representations*, 2021. 3