mathematics

MDPI

Article

Real-Time Motorbike Detection: AI on the Edge Perspective

Awais Akhtar [1], Rehan Ahmed [2], Muhammad Haroon Yousaf [1] and Sergio A. Velastin [3,4,*]

[1] Department of Computer Engineering, University of Engineering and Technology, Taxila 47080, Pakistan; awais.akhtr@gmail.com (A.A.); haroon.yousaf@uettaxila.edu.pk (M.H.Y.)
[2] School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan; rehan.ahmed@seecs.edu.pk
[3] School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK
[4] Department of Computer Engineering, Universidad Carlos III de Madrid, 28911 Leganés, Spain
* Correspondence: sergio.velastin@ieee.org

Abstract: Motorbikes are an integral part of transportation in emerging countries, but unfortunately, motorbike users are also one the most vulnerable road users (VRUs) and are engaged in a large number of yearly accidents. So, motorbike detection is very important for proper traffic surveillance, road safety, and security. Most of the work related to bike detection has been carried out to improve accuracy. If this task is not performed in real-time then it loses practical significance, but little to none has been reported for its real-time implementation. In this work, we have looked at multiple real-time deployable cost-efficient solutions for motorbike detection using various state-of-the-art embedded edge devices. This paper discusses an investigation of a proposed methodology on five different embedded devices that include Jetson Nano, Jetson TX2, Jetson Xavier, Intel Compute Stick, and Coral Dev Board. Running the highly compute-intensive object detection model on edge devices (in real-time) is made possible by optimization. As a result, we have achieved inference rates on different devices that are twice as high as GPUs, with only a marginal drop in accuracy. Secondly, the baseline accuracy of motorbike detection has been improved by developing a custom network based on YoloV5 by introducing sparsity and depth reduction. Dataset augmentation has been applied at both image and object levels to enhance robustness of detection. We have achieved 99% accuracy as compared to the previously reported 97% accuracy, with better FPS. Additionally, we have provided a performance comparison of motorbike detection on the different embedded edge devices, for practical implementation.

Keywords: object detection; motorbike detection; edge devices; real-time; traffic surveillance

MSC: 68T45; 68T07

Citation: Akhtar, A.; Ahmed, R.; Yousaf, M.H.; Velastin, S.A. Real-Time Motorbike Detection: AI on the Edge Perspective. *Mathematics* **2024**, *12*, 1103. https://doi.org/10.3390/math12071103

Academic Editors: Jesús García-Herrero and Johan Debayle

Received: 22 February 2024
Revised: 31 March 2024
Accepted: 4 April 2024
Published: 7 April 2024

1. Introduction

Surveillance systems are a very important and essential part of the built environment due to safety and security. One of the major applications of surveillance is traffic monitoring. Traffic surveillance is an integral part of road safety and security. In terms of security, traffic detection is useful to identify, track, and extract the license plate and behavior of vehicles. Traditional surveillance systems are generally driven by human effort, i.e., a video is captured and people are deployed to examine activity or security risks. However, this approach has disadvantages like high labor costs, lower efficiency, human limitations, and high resource requirements. With the rising trend of artificial intelligence, computer vision and deep learning play a vital role in many advanced applications, ranging from security, medical analysis, sports, military, and general surveillance. These techniques can replace traditional monitoring systems with Intelligent Surveillance Systems (ISSs). This technique takes the input from a camera, i.e., fixed CCTV, drones, or any other imaging source, and helps to identify, track, or classify objects using deep learning algorithms.

For traffic monitoring, road vehicles can be categorized into two main classes: two-wheeled vehicles, and four-wheeled and above vehicles. In developed countries, cars are the most popular means of urban transportation, but in developing nations, the most popular and common means of transportation is motorbikes, mainly because they tend to be more affordable and more easily accessible. However, motorbikes are also more likely to result in fatal road accidents. The World Health Organization (WHO) declared the motorbike as a vulnerable road user (VRU). Motorbikes are one the most venerable road users (VRUs) and are engaged in a large annual number of accidents; therefore, traffic monitoring systems need to detect them in real-time. Once detected, further processing might involve tracking, the detection of rule violations, helmet-wearing compliance, license plate reading, overloading, etc.

Most of the work that has been carried out on road surveillance is for four-wheelers or over (cars, trucks, etc.). Although motorbike detection is equally important, it is much more difficult due to their smaller size, higher occlusion (e.g., often due to the high density of bikes), varying angles of capture, and rider variations. These factors have a high impact on learning algorithms that lead to difficulty in detection. Furthermore, a major difficulty is the unavailability of large datasets. Previous works that have been carried out for motorbike detection have mainly focused on improving the accuracy of bike detection against issues such as occlusion and environmental variations, but they are impractical because of high computation costs, which makes them difficult to deploy in practice. So, the potential challenge that we have identified is in determining how we can make a neural network-based motorbike detection system more cost-efficient for real-time and practical implementation, using off-the-shelf embedded hardware. Regarding bike number plate recognition, it is worth noting that OCR (Optical Character Recognition) is a thoroughly researched technique, as evidenced by numerous studies, e.g., [1–3]. While we recognize the potential advantages of integrating number plate detection with OCR capabilities into our research, our primary objective in this study was to enhance motorbike detection using deep learning models specifically designed for edge devices that can be a precursor to OCR-based plate recognition and association to a specific motorbike.

For real-time implementation, a brute force approach with powerful hardware can complete the work, but another critical challenge is how to make the detection system practical. So, an ideal system should be a small form factor embedded solution for using the model in a real-world scenario. Both solutions are shown in Figure 1.

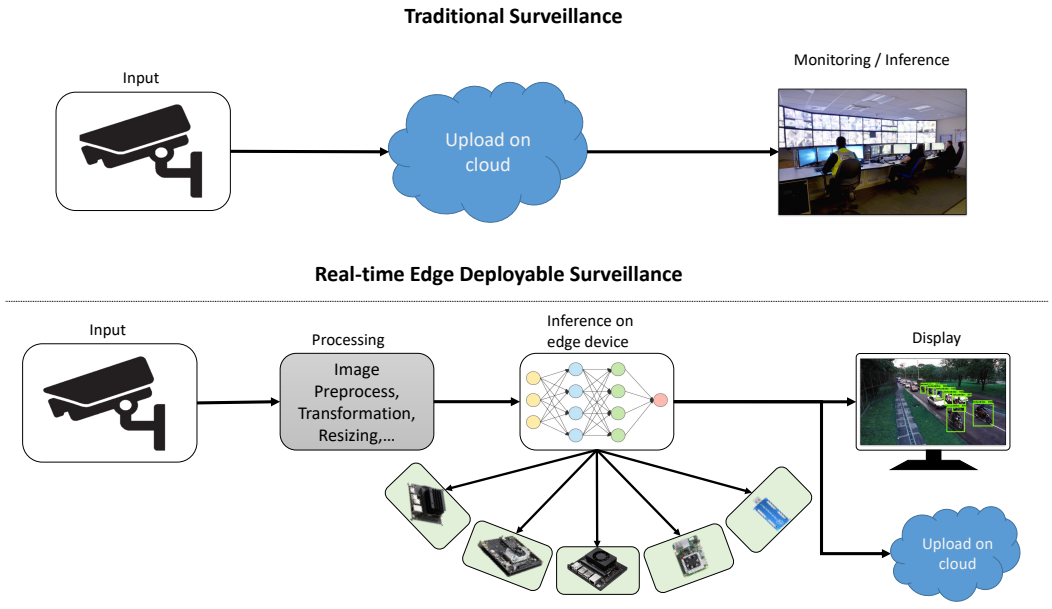

Figure 1. Comparison between traditional surveillance methods and edge deployable surveillance solutions.

This work has studied multiple cost-effective solutions for motor bike detection in real-time as deployable solutions and provides a performance analysis of popular object detection models for this task on different embedded hardware. It has the following contributions:

Improved Baseline Accuracy: Achieved 99% accuracy with custom YoloV5 and augmented dataset, surpassing previous results.

Optimization Analysis: Conducted a comprehensive analysis, comparing motorbike object detection performance on various embedded edge devices, benchmarking results and optimizations.

Edge Deployable Real-Time Systems: Developed real-time motorbike detection models on GPUs, optimized for deployment on NVIDIA, Intel, and Google embedded edge devices.

The rest of the paper is structured as follows: Section 2 describes related work on real-time motorbike detection models for embedded edge devices. Section 3 presents the methodology adopted in this study. Section 4 presents and analyzes the experimental results. Finally, Section 5 provides the conclusions drawn from the study and outlines potential future work.

2. Related Work

Traditionally, motorbike detection has comprised two main steps. Initially, features are detected, followed by classification to distinguish between motorbikes and non-motorbikes, as well as to identify specific features like helmets and license plates. The Circular Hough Transform (CHT) serves as a valuable tool for localizing regions of interest (ROI), as utilized by Silva et al. [4] in helmet detection and by Mukhtar et al. [5] for detecting helmets and headlights on motorbikes. Harr-like features, known for their real-time performance, are employed by Wonghabut et al. [6] and Gavadi et al. [7] in motorbike helmet detection. However, Harr-like features are susceptible to factors such as capture angle and proximity, rendering them less robust for surveillance applications. Moreover, Histogram of Oriented Gradient (HOG) feature extraction is employed by Ghonge et al. [8] to detect license plates of riders without helmets, and also by Dahiya et al. [9] and Singh et al. [10] for a comparative analysis of feature extraction techniques in motorbike and helmet detection. At the time of these studies, HOG was considered one of the most effective feature extractors, alongside SWIFT [11]. In terms of classification, various methods such as Support Vector Machine (SVM), decision trees, random forests, and k-nearest neighbor (KNN) algorithms are employed. Mukhtar et al. [5] and Barics et al. [12] explored SVM classifiers. Shuo et al. [13] utilized SVM with Harr-like features. In contrast, Dupuis et al. [14] employed decision trees for motorcycle classification, combined with manually labeled blobs to mitigate overfitting. Le et al. [15] enhanced classification accuracy by combining random forests with other methods to classify different motorbike components. Waranusast et al. [16] employed K-nearest neighbors (KNNs) for motorcycle classification and helmet detection, while Barics et al. [12] integrated classifiers with hybrid camera systems to achieve improved accuracy.

The paper [17] employs a deep learning approach in conjunction with traditional methods for motorbike detection. They combine a Convolutional Neural Network (CNN) with Histogram of Gradient (HOG) and Support Vector Machine (SVM) for classification and detection. To address false detections, they categorize data into four classes, achieving 84% precision. However, their accuracy significantly decreases in occluded scenarios, and the processing time of two minutes per image renders it unsuitable for real-time implementation. Occlusion poses a significant challenge in motorbike detection. In [18], the authors introduce the annotated dataset MB7500 to handle occlusion, employing a Faster R-CNN-based network with a custom two-layer CNN. Despite these efforts, detection remains irregular, with accuracy dropping to 20% in heavily occluded scenes. The low frame rate per second (fps) of Faster R-CNN makes real-time application challenging. In [19], a four-layer CNN feature extractor serves as the backbone for Faster R-CNN, augmented with the Markov Decision Process (MDP) for tracking on the MB1000 dataset, achieving 88% mean average precision (mAP) in occluded scenarios. However, the compute-intensive two-step detection process results in high inference time, making it impractical for real-time deploy-

ment, especially on edge devices. Addressing false detections, missed detections, and small objects with complex backgrounds, the authors od [20] propose a customized technique alongside a Fast R-CNN-based deep neural network. Despite achieving high accuracy, reaching 90.8% and 88.6% for low and high difficulty sets, respectively, the computational costs are considerable, prioritizing detection robustness over real-time applicability. In [21], Faster R-CNN with Inception-ResNet backbone performs well, with the single-stage SSD network with Inception outperforming others in accuracy with preprocessing techniques. However, real-time applications are not considered, posing challenges for embedded solutions. The work in [22] utilizes deep learning for traffic surveillance, extending to detect traffic violations. They incorporate the "DhakaAI" dataset with their images, enhancing it with 17 vehicle types, including around 4200 motorbike images. While achieving 15–20 fps using YoloV4 and 24–30 with 72.02% mAP using YoloV4-tiny, real-time deployment on edge devices is hindered by low mAP and fps. In [23], a custom network based on SSD is proposed for detecting complex traffic scenarios, achieving an mAP of 89.05% and an overall speed of 32 fps. While meeting real-time criteria, challenges persist in deployment on embedded platforms. Proposing a methodology for motorbike rider detection with and without helmets, the paper [24] trains two models using Faster R-CNN with the Inception V2 model, achieving 93.37% accuracy overall. However, the fps is not discussed, suggesting potential difficulties in real-time operation and deployment on embedded platforms.

Regarding datasets, detecting two-wheeled vehicles presents a challenging task, primarily due to the scarcity of comprehensive datasets. Despite the critical nature of this issue, there exists no single dataset that sufficiently covers all aspects and variations essential for benchmarking two-wheeled vehicle detection. This lack of availability of a comprehensive dataset has been underscored by the authors of [25]. One contributing factor to this scarcity could be the smaller contribution of motorbikes to traffic in developed nations compared to the developing world. Several benchmark datasets include the motorbike category, albeit with various limitations. For instance, the dataset introduced in [26] comprises five videos and 3880 annotated images, encompassing diverse traffic scenarios. However, it lacks motorbike category annotations and is more suited for autonomous car applications rather than wide-area surveillance. Similarly, the CBCL street scenes database [27], containing 8000 images with annotations across nine object classes, is relevant for street scene understanding but lacks motorbike images in urban settings. KITTI [28], renowned in autonomous vehicle research, offers datasets for optical flow, stereo, 3D object detection, tracking, and semantic segmentation. Despite its versatility, KITTI does not include a motorbike category. The PASCAL VOC dataset, a popular benchmark for detection, segmentation, and classification tasks, includes the motorbike category but with only 713 annotated images by 2012 [29,30]. Its limited size raises concerns about overfitting, and its focus on detection makes it less suitable for surveillance applications. The Caltech dataset [31] features 30,607 images spanning 256 object categories, including the motorbike category with 798 images. However, its limitation lies in the absence of frontal views, rendering it less suitable for surveillance purposes. Some researchers resort to custom datasets, which unfortunately are not publicly available. These include datasets used by [4,13,14,32–37]. The public MB7500 dataset, published by the authors of [38], stands out for its coverage of occlusion and representative surveillance viewpoints, making it a valuable resource for two-wheeled detection datasets. To encapsulate the diverse approaches employed by various authors in motorbike detection, it is evident that different methods and models have been utilized. Among these, the approach outlined by the authors of [19] stands out prominently, boasting an exceptionally high mean average precision (96%). This work suggests that SSD Mobilenet emerges as the optimal choice for motorbike detection among available methods. Shifting the focus to dataset comparisons, it becomes apparent that MB7500 holds a position of prominence. This dataset stands out as the most suitable for motorbike detection, particularly from a surveillance perspective. Not only is MB7500 publicly accessible, but it also stands as the most frequently utilized dataset for motorbike detection in prior research publications.

3. Methodology

The main purpose of this work is to investigate motorbike detection using state-of-the-art neural networks on low-power, embedded edge devices in real-time. Several steps are applied to accomplish this task. A detailed flow of all these steps is depicted in Figure 2. In the first step, a dataset is chosen and prepared for the application. Secondly, an appropriate model is selected and trained. Lastly, a suitable edge device is selected and model optimization is applied to generate the final inference model file for deployment. All these steps are discussed in more detail in the following sections.

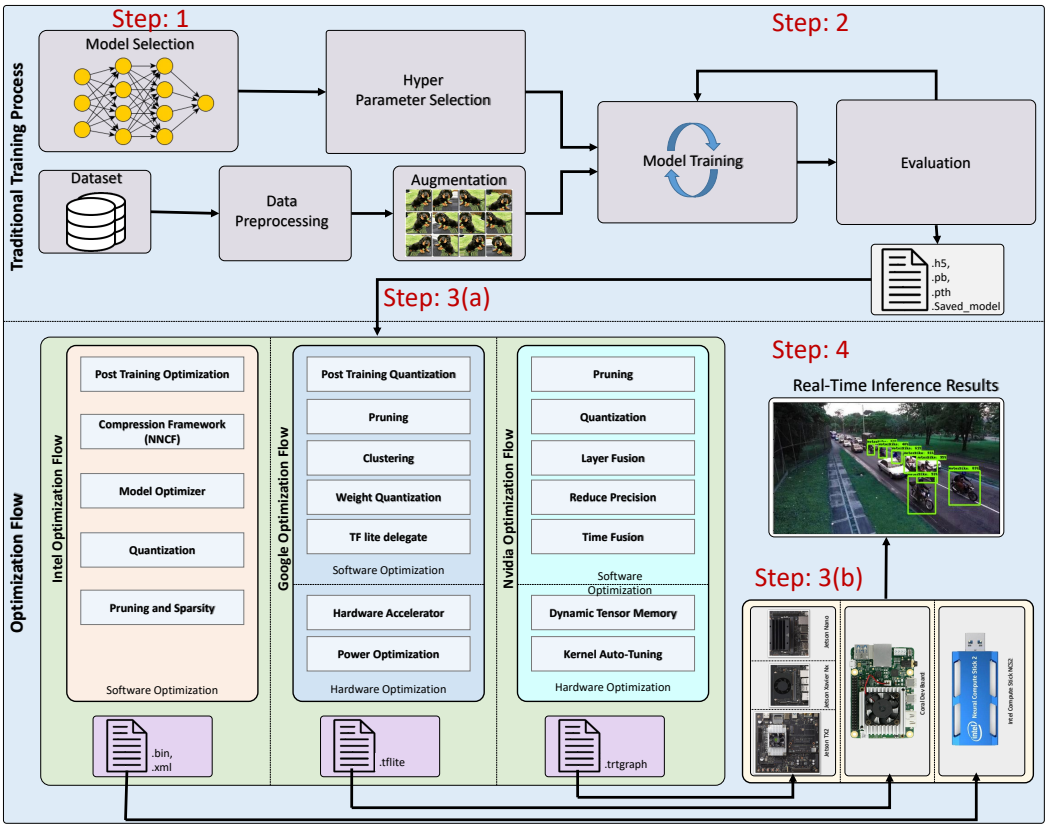

Figure 2. Optimization flow for deploying a deep learning algorithm on edge devices: illustrating the process of tailoring a model for efficient execution on edge devices.

3.1. Dataset and Augmentation

The performance of a deep learning model heavily depends upon the quality and quantity of the dataset. As discussed earlier, for motorbike detection, MB7500 is the most suitable current dataset. However, it has some issues, such as a lack of variability, i.e., lighting condition, angle of capture, and other environmental conditions that are not covered in the dataset. To enhance the quantity and introduce variability in the dataset, image level and object level augmentation are performed. Image level is a type of augmentation that is applied on the entire image and object level augmentation is applied only within the object bounding box. In total, 14 different types of augmentations are applied that include flip, crop, 90°rotation, minor rotations, grayscale, saturation, brightness, blur, and mosaic for image level and flip, crop, 90°rotation: clockwise and anticlockwise, minor rotations, and brightness for object level augmentation. As a result, the dataset size is increased from 7500 images to 15,000 images. Figure 3 depicts the augmentation types.

Figure 3. Augmentation techniques applied at image and object levels: depicting crop, flip, rotate, and brightness adjustments for enhanced object detection performance.

Augmentations not only help to increase the dataset but also help introduce a more difficult scenario to improve accuracy on the different datasets. To make the model less sensitive to camera orientation, roll-robustness, position, and subject translation are performed by rotation, flip, 90° rotation, rotation of the frame (30° rotation in this case), and by cropping the frames randomly.

3.2. Model Selection

Motorbike detection is an object detection task with several models available for this purpose, as pointed out earlier. Paper [21] reports several experiments with over 50 networks to identify the best model for motorbike detection and concludes that SSD with a Resnet backbone works best for this application. This work also uses the SSD architecture, but its Resnet backbone is replaced with Mobilenet V2. This is because Resnet can only perform on a GPU-based system and is not suitable for edge devices, while Mobilenet V2 is specialized for edge devices. Then, since motorbike detection is a surveillance application and Yolo models are well suited for this task and are suitable for edge devices, we also studied the use of YoloV5 for this task.

3.2.1. Single Shot Detection (SSD)

Mobilenet V2 backbone acts as a feature extractor; it is comparatively faster than other backbone architectures and is a popular choice for real-time applications on edge devices. For this reason, SSD Mobilenet V2 is supported by all of the major edge devices' optimization engines. In this work, the SSD Mobilenet V2 with the augmented images of the MB7500 dataset has been trained for 1000 epochs on a GPU.

3.2.2. Customized YoloV5

For most surveillance applications, Yolo architectures are used. Not too long ago, YoloV5 was released which outperformed all its predecessors. YoloV5 offers the flexibility of customizing the architecture according to the available computation resources, which can be very advantageous for real-time applications and edge deployment. Another advantage of YoloV5 is that it is known to generate very stable inference results in traffic surveillance with high mAP. The architecture used here is customized according to the computation resources on the edge devices so that it can perform in real-time. To reduce the computation of YoloV5, layer reduction and sparsity techniques have been applied.

For a fair comparison, the same dataset (Augmented MB7500) has been used for both networks. To train these two networks, transfer learning has been applied to further boost accuracy. This has been performed using pre-trained weights of these models for motorbike detection. These networks have been trained iteratively with different hyperparameters until the best accuracy is achieved in terms of mAP. After training on a GPU, we achieved 91.5% mAP for SSD Mobilenet V2 and 99% mAP for YoloV5. This sets the baseline performance for the optimization process, as shown in Table 1.

Table 1. Baseline performance comparison of SSD Mobilenet V2 vs. YoloV5.

Network	Mean Average Precision (mAP)	Inference Time (ms)	Frames Per Second (FPS)
SSD Mobilenet V2	91.5%	20.218	49.46
YoloV5	99.0%	10.526	94.0

3.3. Edge Deployment

This section is divided into two main parts. First, it deals with the process of selecting suitable edge devices. Secondly, it explores the optimization of algorithms to ensure efficient execution on the chosen edge devices.

3.3.1. Edge Devices

Conventional detection procedures are not generally deployable solutions for real-time applications, due to extensive computation resource requirements and high power consumption. So, an appropriate edge device has to be selected for implementing a model in real-time. Edge devices come in all shapes and sizes with their respective advantages and disadvantages. Currently, there are many options available to choose from, but the main three types of edge devices are Intel, Nvidia, and Google. Before deploying a model on the edge device, it needs to pass through a vendor-specific optimization tool that reduces the computation cost and inference time of the model. Below is a list of available edge devices from these three suppliers:

1. Google (USA, CA, Mountain View)
   - Coral Dev Board.
2. NVIDIA (USA, CA, Santa Clara)
   - Jetson Nano;
   - Jetson Tx2;
   - Jetson Xavier Nx.
3. INTEL (USA, CA, Santa Clara)
   - Intel Neural Compute Stick 2 (NCS2).

This work uses all these devices so that the best platform can be identified. The specifications of each of these devices are shown in Table 2 and the devices are shown in Figure 4.

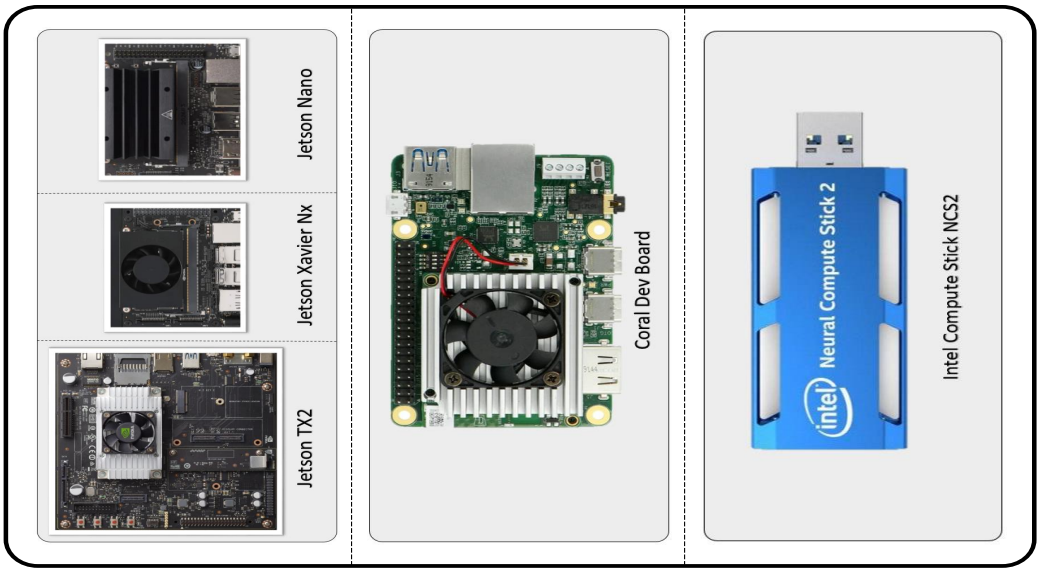

Figure 4. Various embedded devices utilized in the study for real-time deployment of deep learning models.

Table 2. Comparison of edge devices: detailed comparison of hardware components and features across various embedded platforms.

Specification	Jetson Nano	Jetson TX2	Jetson Xavier Nx	Coral Dev Board	NCS2
AI Performance	472 GFLOPs	1.26 TFLOPs	21 TOPs	4 TOPS	1 TOPS
RAM	4 GB DDR4	8 GB DDR4	16 GB DDR4	1GB LPDDR4	-
Flash Memory	16 GB eMMC 5.1	32 GB eMMC 5.1	16 GB eMMC 5.1	8 GB eMMC	-
CPU	Quad-core ARM Cortex A57	(1) Quad-core ARM Cortex A57 (2) Dual-core Nvidia Denver2	Octa-core ARM V8.2	Quad-core ARM Cortex-A53	-
Clock Speed	1.43 GHz	2 GHz	2.265 GHz	1.5 GHz	700 MHz (Processor Base Frequency)
Accelerator Hardware	128 Nvidia Maxwell GPU	256 Nvidia Pascal GPU	(1) 512-Nvidia Volta GPU with 64 Tensor Cores (2) 2 × NVDLA v1	Google Edge TPU Accelerator	Intel Movidius Myriad × VPU 4GB
Operating Systems	Linux4Tegra	Linux4Tegra	Linux4Tegra	Debian Linux	(1) OS Independent (2) OpenVINO toolkit
Power Required	5–10 watt	7.5–15 watt	10–15 watt	10–15 watt	-

3.3.2. Optimization

The conventional deep learning and machine learning algorithm development and deployment cycle has two phases. The first phase is the training phase in which the focus is on selecting a suitable neural network architecture and dataset preparation to train the model. In this process, the priority is on the desired accuracy because the higher the validation accuracy, the better the solution. The second phase is the inference process where the trained model is used to predict unseen data and the goal is to achieve comparable accuracy in making a decision as fast as possible. Conventional approaches use brute force to increase the computation resources to make this possible. A more suitable approach is to transform the trained model such that it uses less computation and resources, this process is called optimization. Optimization offers several advantages as it helps to reduce the size of the network and uses low precision to make it possible to efficiently run large-scale networks on edge devices. Optimization is what stands between conventional inference and a smart, compact, edge deployable deep learning solution. Due to this, optimization has now become the third phase of the modern machine learning-based solution development cycle.

Optimization is a general strategy that can be performed directly on the model but where each vendor provides their proprietary tools to optimize the model. This is because the optimization of a model for an edge device heavily depends upon the device architecture, and vendors do not expose the architectural details of their devices. TensorRT (Release 9.3.0) is used for optimizing the trained model for Nvidia's edge devices. OpenVINO (Release 2023.3.0) is an Intel open-source toolkit for the optimization of the network to be used for inference and quick deployment. TensorFlow Lite (Version 2.9) is an open-source package of deep learning tools/framework to carry out the optimized deep learning inference on the edge for Google Edge devices. In general, there are two types of optimization: software-related optimization and hardware-related optimization. Each optimizer uses different methods to optimize the deep learning trained graph, but the general sub-processes that are involved in the optimization process are quantization, pruning, layer fusion, clustering, sparsity, time fusion, kernel auto-tuning, dynamic tensor memory, and so on.

Since we are using all these devices, all these optimization tools have been used in this work. Optimization is not a stable process and it often generates uncertain outputs (low

accuracy, high inference time, high memory consumption, etc.), which requires iterative adjustments of user-defined parameters to achieve the best results.

4. Results

The final optimized network file is copied to the target device to run it for producing predictions. Then, the performance of the optimized network can be measured on the target device. If the desired results are achieved, the solution can be applied in the field. Before proceeding with the results, it is necessary to understand the evaluation metrics, as discussed next.

4.1. Experimental Setup

In this work, the object detection system implemented on the edge devices is coded in Python due to its versatility, ease of development, and compatibility with the selected frameworks and libraries. These choices were made to facilitate comprehensive comparison of the performance of state-of-the-art edge devices with that of a GPU, considering metrics such as mAP (mean average precision), FPS (frames per second), model size after optimization, power consumption, and memory usage. Additionally, two baseline models have been established to mitigate any bias stemming from a particular device on a model.

4.1.1. Dataset

As pointed out before, there is a general lack of suitable datasets to train motorbike detection models [25]. Although no comprehensive dataset is available, we have selected the MB7500 open-source dataset, contributed by Espinosa et al., because it is focused on motorbikes with surveillance-like views. Although it would have been possible to use KITTI, it is mainly aimed at autonomous driving and mobile robotics as the camera is mounted on a car, facing forward. Nevertheless, it also does not include a separate class of motorbikes.

The data was collected for MB7000 using a Phantom 4 drone with a video camera. The dataset consists of 7500 annotated images of motorbikes in urban occluded scenarios with a size of 640 $\times$ 364, including 60% of occluded scenarios. The dataset is split into a 60–40 ratio to avoid overfitting. The dataset is collected using a drone with an angle similar to a CCTV camera. This dataset covers the occluded scenario very efficiently. However, this dataset does not address the problem of diversity, i.e., lightning conditions, weather conditions, angle problems, and the diversity in shape, place, and number of riders.

4.1.2. Evaluation Metrics

In a normal classification-based task, the performance of a network is defined in terms of accuracy, and this is sufficient to justify the impact of the method. However, in an object detection-based task, performance is better evaluated based on mAP, which further depends on ROI (region of interest) and prediction accuracy. The actual region of interest is the area on the image where the actual object exists. The area on the image that is predicted by the model in which it believes that some object is identified can be referred to as the predicted region of interest. A correct detection is measured as the overlap (typically 50%) between the predicted region of interest and the actual region of interest, given that the object identified in that area is the same as the actual label. This is the performance criteria for object detection-based tasks. To evaluate the performance of an object detection model in real-time, further metrics are used such as frames per second or inference time per frame. However, further metrics are required to demonstrate the performance of such object detection-based tasks on small, embedded edge devices. For that, we are using model size, power consumption, and memory usage, where the latter two are monitored during the runtime. This gives us insight into the real-time performance of different devices against different optimization flows, for the common task and neural network architecture.

4.2. Result and Analysis

This section presents a detailed analysis of the performance of SSD Mobilenet V2 and YoloV5 models for real-time motorbike detection on various embedded edge devices and a standard GPU. Although YoloV8 is a recent introduction, potential deployment in the field is limited due to licensing costs, and we selected YoloV5 for comparison due to its relevance to our research objectives and full availability at the time of the study. We begin by measuring the baseline performance of SSD Mobilenet V2 on a normal GPU, serving as a reference for comparison. This section highlights the mean average precision (mAP) scores and frames per second (FPS) achieved by each device, offering insights into their computational power and real-time capabilities. Additionally, we explore the impact of model size on the trade-off between accuracy and speed for edge devices. Furthermore, we present inference results of SSD Mobilenet V2 on both the GPU and edge devices, showcasing the detection performance with predicted bounding boxes. Then, we present the evaluation of YoloV5 with and without dataset augmentation, along with various input tensor sizes. A comparison of YoloV5 with other state-of-the-art models and custom networks is provided based on parameter count, mAP, and FPS on the GPU. Overall, the results reveal the advantages and drawbacks of utilizing these models for real-time motorbike detection tasks on different embedded edge devices and a standard GPU.

4.2.1. Results and Analysis Using SSD Mobilenet V2

First, we measure the performance of this model on a normal GPU, which gives the baseline performance. From Figure 5, it can be seen that the Single Shot Detection (SSD) Mobilenet V2 gives a 91% mean average precision score, which is higher than all the embedded edge devices but at the cost of high computation power and memory, as shown in Figure 6. On the other hand, the GPU processes around 49 FPS, which is more than the real-time requirement but is somewhat low compared to most edge devices, given the fact that edge device are very low-power and limited on resources (Figure 5).

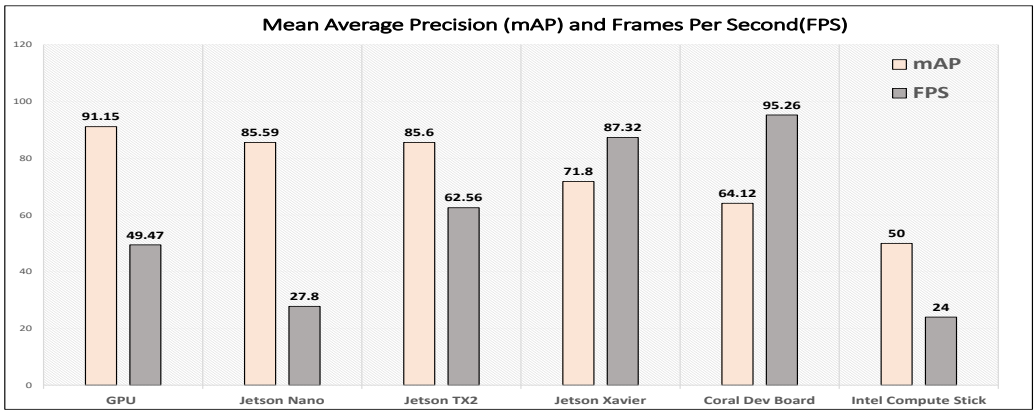

Figure 5. Speed and accuracy comparison across different devices for SSD Mobilenet V2: visualizing performance metrics of various edge devices for real-time deep learning model deployment.

As far as mAP and FPS are concerned, it can be observed from Figure 5 that the mean average precision score of the edge devices starts to decrease with an increase in FPS. Therefore, the devices offer a straight trade-off between FPS and mean average precision. To maintain a certain FPS and real-time performance, a device has to drop its mAP. This way, despite their small sizes and limited resources, some of these devices can exceed the GPU performance in terms of FPS score, i.e., Coral Dev Board achieves an FPS twice as high as THE GPU, Jetson Xavier achieves 80% higher FPS as compared to the GPU, and Jetson TX2 achieves 63 FPS, which is also considerably higher than the 49 FPS of the GPU.

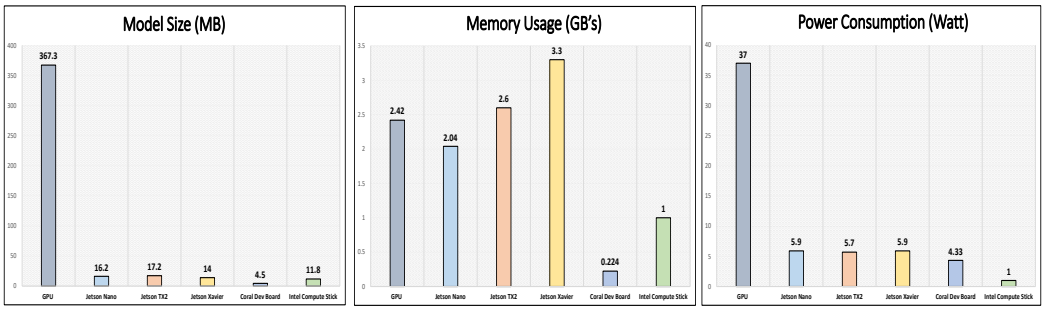

Figure 6. Comparison of model size, memory usage, and power consumption across different devices for SSD Mobilenet V2: visualizing resource utilization metrics for real-time model execution on various edge devices.

On the other hand, from Figure 5, it can be seen that some smaller edge devices lack performance in terms of mean average precision (Intel compute stick); moreover, although their FPS is lower than that of the GPU (Intel compute stick and jetson nano), they are closer at reaching the threshold of real-time performance.

When looking at the impact of model size, since we are using the same model for the evaluation of the performance on all edge devices, the optimized model size can provide a direct measure of the effectiveness of the model optimization flow of a respective vendor that has caused a significant reduction in the computation complexity of a model. One core observation, from Figure 6, is that a reduction in the computation complexity of a model causes a decrease in mAP while it increases the FPS. Optimization of the Coral Dev Board provides the highest compression in terms of model size, and this in turn provides the highest frame rate.

Inference results of different frames of SSD Mobilenet V2 on GPU are shown in Figure 7. These are the same set of images as depicted during dataset selection but with predicted bounding boxes. For comparison, inference results of embedded devices are also presented in Figures 8–10 for TX2, Coral Board, and Xavier, respectively. It can be observed that the highest number of bikes are detected in the GPU-based implementation, but almost the same number of bikes are detected on the edge device-based implementations as well, and with higher FPS.

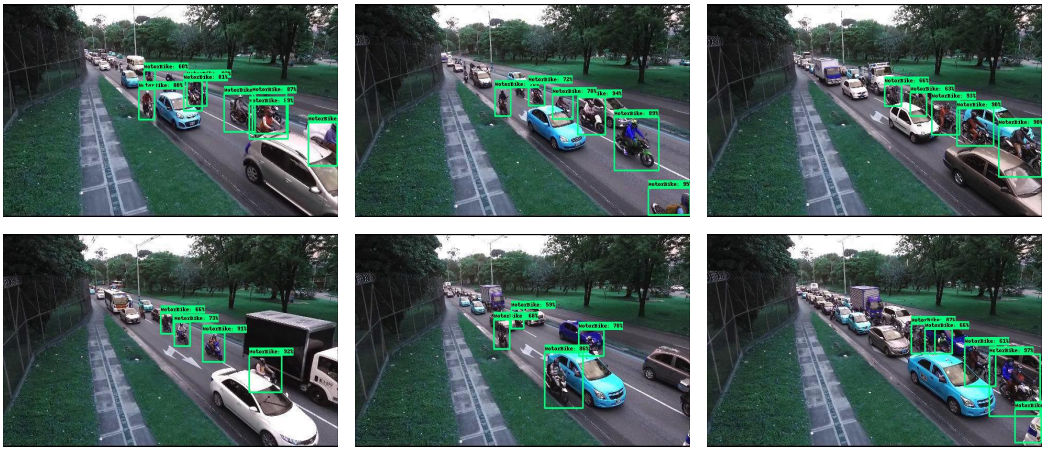

Figure 7. Inference results of SSD Mobilenet V2 on GPU: visualization of object detection performance using GPU acceleration.

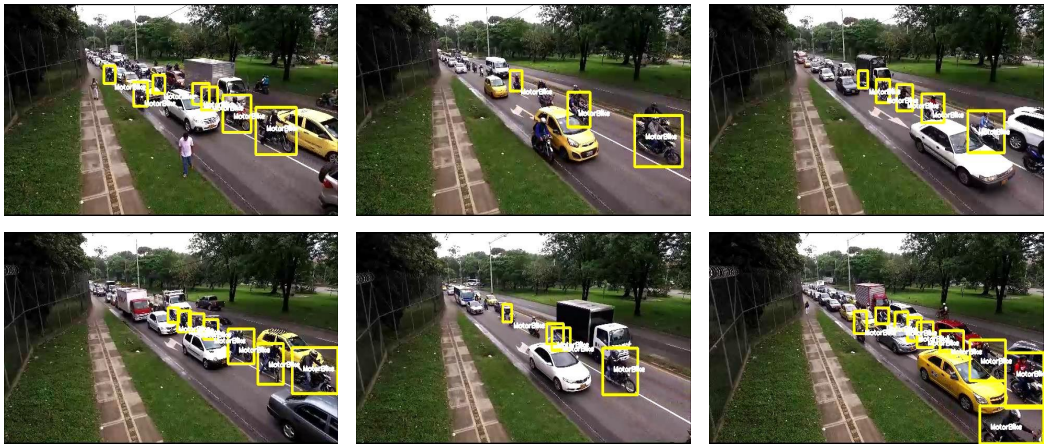

Figure 8. Inference results of SSD Mobilenet V2 on Jetson TX2 (NVIDIA): object detection performance on Jetson TX2.

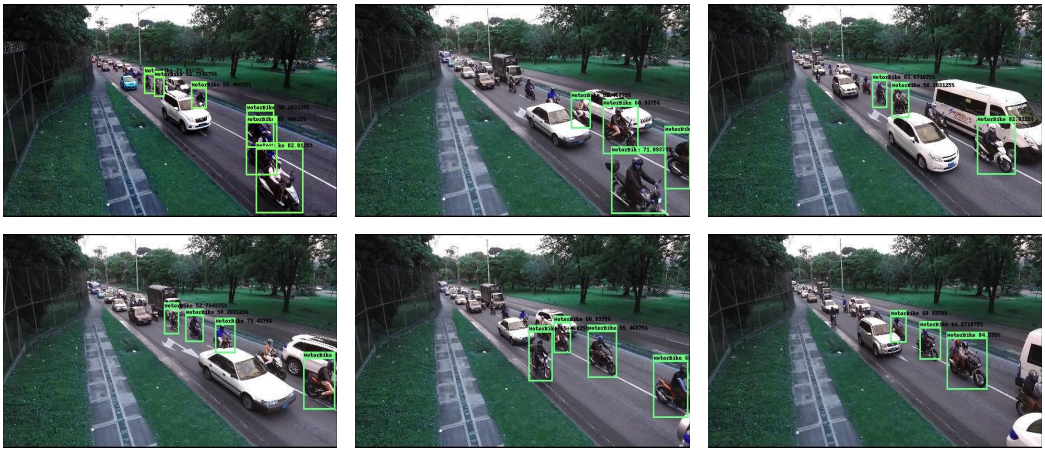

Figure 9. Inference results of SSD Mobilenet V2 on Coral Dev Board (Google): object detection performance on Coral Dev Board.

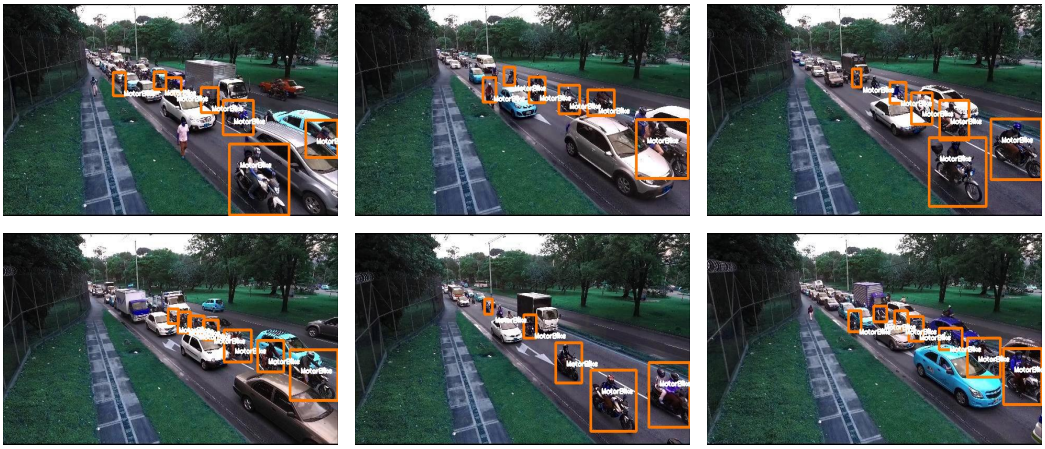

Figure 10. Inference results of SSD Mobilenet V2 on Jetson Xavier (NVIDIA): object detection performance on Jetson Xavier.

4.2.2. Results and Analysis of Customized YoloV5

YoloV5 has been trained with two settings, i.e., with and without dataset augmentation and also with different-sized input tensors. The results of YoloV5 on a GPU and the Jetson Xavier are presented in Table 3. It can be seen that training without augmentation gives around 97 percent mAP, while training with augmentation gives around 99 percent mAP, while in terms of FPS, it gives around 65 FPS on 640 × 364-sized images. When the image size is decreased to 300 × 300, we see some drop in mAP, but the inference rate goes up to over 100 FPS.

Table 3. Results of YoloV5 on GPU and edge devices: object detection performance comparison (bold indicates best performance).

Device	Augmentation	Input Size	mAP (%)	FPS	Memory (GB)	Power (W)
GPU	No	640 × 364 × 3	**97**	**65**	1.4	45
		300 × 300 × 3	94	103	1.2	45
	Yes	640 × 364 × 3	**99**	**65**	1.4	45
		300 × 300 × 3	95	103	1.2	45
Xavier	Yes	640 × 364 × 3	90	27	3.3	5.8
		300 × 300 × 3	86	**51**	2.7	5.1

YoloV5 provides around 90 percent mAP on a Jetson Xavier with 27 FPS. This low FPS is due to partial optimization and these results can be further improved with optimal optimization as discussed earlier. There is, however, another way to improve the FPS, i.e., image size reduction. A model has been trained with a 300 × 300 image size, and when this small network is partially optimized for Jetson Xavier, the FPS increases to 51 FPS. This significant boost is due to reduced computation, caused by a small frame size with only a marginal drop in mAP. This suggests that YoloV5 can still be used with partial optimization on a Jetson Xavier with acceptable mAP and FPS. For comparison, the inference results of different frames of YoloV5 on the GPU and on Jetson-Xavier are shown in Figures 11 and 12, respectively.

Table 4 provides a comparison of different variants of YoloV5, along with a custom network, based on parameter count, mAP and FPS on a standard GPU. It can be seen that the custom network provides almost the same mAP but has a significantly reduced parameter (size) count.

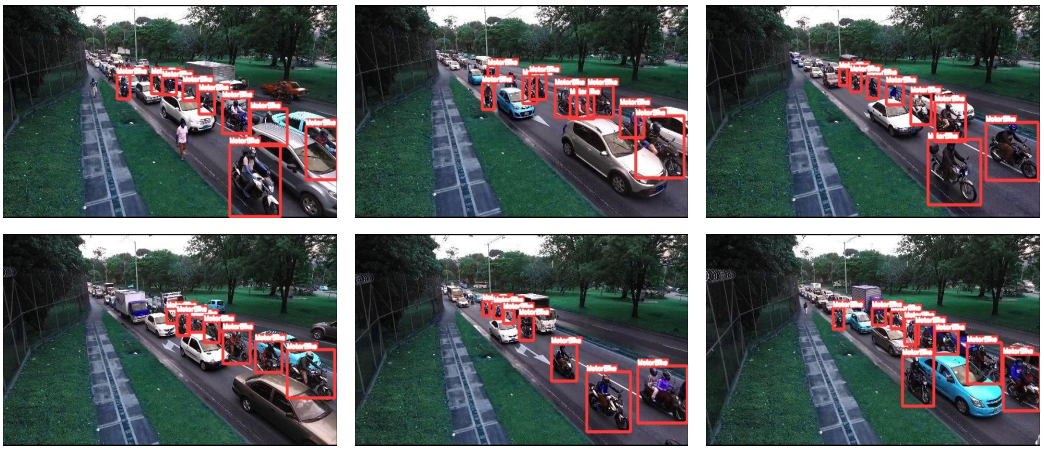

Figure 11. Inference results of YoloV5 Model on GPU: object detection performance using GPU acceleration.

Figure 12. Inference results of YoloV5 on Jetson Xavier (NVIDIA): object detection performance on Jetson Xavier.

Table 4. Comparison of different variants of YoloV5 with custom model: performance evaluation of various YoloV5 configurations.

Model	Parameters	GFLOPS	mAP
YoloV5 Extra Large	87.7 M	218.8	99.458
YoloV5 Large	47.0 M	115.4	99.447
YoloV5 Medium	21.4 M	51.3	99.437
YoloV5 Small	7.3 M	17.0	99.434
YoloV5 Custom	32 k	1.0	98.975

Table 5 provides a comparison of YoloV5 with other state-of-the-art models used in previous papers. As mentioned earlier, we have trained YoloV5 with and without dataset augmentation. From Table 5, it can be seen that training without augmentation gives around 97% mAP while training with augmentation gives around 99% mAP. Both these accuracies are better than all other previous baseline accuracies. On the other hand, we obtain around 65 FPS on the GPU with YoloV5, which is also better than the previously reported inference rates. In conclusion, using YoloV5 has the following advantages:

- It gives consistently high mean average precision.
- It gives higher FPS on GPU.
- Model size is very small.
- It uses less memory resources.

Table 5. Comparison of YoloV5 with other models: performance evaluation across different object detection models.

Model	Input Size	mAP	FPS
YoloV3	$640 \times 364 \times 3$	0.89	21
SSDLite Mobilenet V2	$300 \times 300 \times 3$	0.96	60
SSD InceptionV2	$300 \times 300 \times 3$	0.94	60
SSD Mobilenet V2	$640 \times 364 \times 3$	0.97	60
YoloV5	$640 \times 364 \times 3$	H	H

One drawback of YoloV5 is that its robustness is low compared to SSD Mobilenet. In our observation, SSD Mobilenet trained on one motor bike dataset can be used for other motor bike datasets as well, but YoloV5 does not have such robustness.

5. Conclusions and Future Work

In this paper, we have presented an investigation of motorbike detection on edge devices for real-time applications. We have investigated five different state-of-the-art

embedded edge devices from three different vendors, using state-of-the-art SSD Mobilenet and YoloV5. The most important aspect that differentiates between conventional machine learning implementation and edge deployable machine learning is the optimization of the neural network. We have used three different frameworks/toolkits to perform optimization for edge devices. We have demonstrated that with proper optimization, neural networks can be made to run on small, low-power edge devices with some pros and cons. Firstly, model size is significantly reduced by several folds after optimization. Due to this reduced model size, power consumption and memory usage drop significantly. Reduced model size is a direct outcome/indication of reduced computation complexity, which in turn increases inference rate. For some embedded devices, we have achieved inference rates that are twice as high as a GPU. For the Coral Dev Board, the inference rate is 96 FPS; for Nvidia Xaviera, the inference rate is 88 FPS; and for Nvidia Jetson TX2, the inference rate is 63 FPS, which is higher than the standard GPU inference rate of 53 FPS. A boost in inference rate comes at the cost of some decrease in mAP. In this case, this decrement lies in the range of 5 percent of the original baseline accuracy, which is acceptable for most edge devices. On the other hand, we have improved the previous best baseline accuracy of motorbike detection using a custom network based on YoloV5 on a standard GPU-based system. We have achieved 99 percent mAP with 65 FPS, which is around 2–3 percent better than the previous mAP and 15 FPS higher.

One of the possible future directions is related to the unavailability of the dataset. We have used the MB7500 dataset with some custom-labeled images. This is a small dataset in terms of the total number of images. Future work could include collecting much more representative data for motorbike-related analysis, including not only the variability of the motor vehicle, but also the atmospheric conditions, type of riders, and traffic violations. Another useful work would be to investigate and improve the lack of robustness of YoloV5 in "domain shifts", i.e., when data change in type, camera view, etc. This is a major problem that currently requires the time-consuming manual annotation of additional data.

Author Contributions: Conceptualization, R.A.; methodology, R.A., M.H.Y. and S.A.V.; software, A.A.; validation, A.A.; formal analysis, A.A. and M.H.Y.; investigation, A.A.; resources, R.A.; data curation, M.H.Y. and S.A.V.; writing—original draft, A.A.; writing—review & editing, R.A., M.H.Y. and S.A.V.; supervision, R.A. and M.H.Y.; project administration, R.A.; funding acquisition, M.H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: http://videodatasets.org/UrbanMotorbike.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Keerthi, K.V.L.; Krishna Teja, V.; Chandra Sekhar, P.N.R.L.; Shankar, T.N. Automatic Recognition of Helmetless Bike Rider License Plate Using Deep Learning. In *Computational Vision and Bio-Inspired Computing*; Advances in Intelligent Systems and Computing; Smys, S., Tavares, J.M.R.S., Bestak, R., Shi, F., Eds.; Springer: Singapore, 2021; pp. 457–467.
2. Roy, R.; Kumar, S.; Dumbhare, P.; Barde, M. Helmet Detection and Number Plate Recognition using Machine Learning. In Proceedings of the 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 1–3 July 2022.
3. Premmaran, G.; Sathishkumar, P. Detection of Helmetless Riders and Automatic Number Plate Recognition Using Machine Learning. In Proceedings of the 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 9–11 May 2022; pp. 339–345. [CrossRef]
4. Silva, R.R.; Aires, K.R.; Veras, R.D. Detection of helmets on motorcyclists. *Multimed. Tools Appl.* **2018**, *77*, 5659–5683. [CrossRef]
5. Mukhtar, A.; Tang, T.B. Vision based motorcycle detection using HOG features. In Proceedings of the 2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Lumpur, Malaysia, 19–21 October 2015; pp. 452–456.
6. Wonghabut, P.; Kumphong, J.; Satiennam, T.; Ung-Arunyawee, R.; Leelapatra, W. Automatic helmet-wearing detection for law enforcement using CCTV cameras. In *Proceedings of the IOP Conference Series: Earth and Environmental Science*; IOP Publishing: Bristol, UK, 2018; Volume 143, p. 012063.
7. Gavadi, R.J.; Patil, S.S. Automatic detection of motorcyclist without helmet using haar cascade classifier. *J. Integr. Sci. Technol.* **2018**, *6*, 33–36.

8. Ghonge, S.A.; Sanghavi, J.B. Smart surveillance system for automatic detection of license plate number of motorcyclists without helmet. *Int. J. Comput. Sci. Eng* **2018**, *2*, 86–89. [CrossRef]
9. Dahiya, K.; Singh, D.; Mohan, C.K. Automatic detection of bike-riders without helmet using surveillance videos in real-time. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 3046–3051.
10. Singh, D.; Vishnu, C.; Mohan, C.K. Visual Big Data Analytics for Traffic Monitoring in Smart City. In Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016.
11. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; Volume 2, pp. 1150–1157.
12. Barış, İ.; Başţanlar, Y. Classification and tracking of traffic scene objects with hybrid camera systems. In Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, 16–19 October 2018; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2018.
13. Shuo, Y.; Choi, E.J. A driving support system base on traffic environment analysis. *Indian J. Sci. Technol* **2016**, *9*, 286–290. [CrossRef]
14. Dupuis, Y.; Subirats, P.; Vasseur, P. Robust image segmentation for overhead real time motorbike counting. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 3070–3075.
15. Le, T.S.; Huynh, C.K. An unified framework for motorbike counting and detecting in traffic videos. In Proceedings of the 2015 International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, Vietnam, 23–25 November 2015; pp. 162–168.
16. Waranusast, R.; Bundon, N.; Timtong, V.; Tangnoi, C.; Pattanathaburt, P. Machine vision techniques for motorcycle safety helmet detection. In Proceedings of the 2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013), Wellington, New Zealand, 27–29 November 2013; pp. 35–40.
17. Huynh, C.K.; Le, T.S.; Hamamoto, K. Convolutional neural network for motorbike detection in dense traffic. In Proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha-Long, Vietnam, 27–29 July 2016; pp. 369–374.
18. Espinosa, J.E.; Velastin, S.A.; Branch, J.W. Motorcycle detection and classification in urban Scenarios using a model based on Faster R-CNN. In Proceedings of the 9th International Conference on Pattern Recognition Systems (ICPRS 2018), Valparaíso, Chile, 22–24 May 2018.
19. Espinosa, J.E.; Velastin, S.A.; Branch, J.W. Detection and tracking of motorcycles in congested urban environments using deep learning and Markov decision processes. In Proceedings of the Mexican Conference on Pattern Recognition, Querétaro, Mexico, 26–29 June 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 139–148.
20. Wang, K.; Zhou, W. Pedestrian and cyclist detection based on deep neural network fast R-CNN. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419829651. [CrossRef]
21. Kausar, A.; Jamil, A.; Nida, N.; Yousaf, M.H. Two-wheeled vehicle detection using two-step and single-step deep learning models. *Arab. J. Sci. Eng.* **2020**, *45*, 10755–10773. [CrossRef]
22. Shubho, F.H.; Iftekhar, F.; Hossain, E.; Siddique, S. Real-time traffic monitoring and traffic offense detection using YOLOv4 and OpenCV DNN. In Proceedings of the TENCON 2021–2021 IEEE Region 10 Conference (TENCON), Auckland, New Zealand, 7–10 December 2021; pp. 46–51. [CrossRef]
23. Miao, Y.; Zhang, S.; He, S. Real-Time Detection Network SI-SSD for Weak Targets in Complex Traffic Scenarios. *Neural Process. Lett.* **2022**, *54*, 3235–3247. [CrossRef]
24. Khandelwal, Y.; Anwar, S.; Agarwal, S.; Tripathi, V.; Pandey, P. A Framework for Enhancing the Security of Motorbike Riders in Real Time. In *Intelligent Computing in Engineering*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 275–281.
25. Espinosa, J.E.; Velastín, S.A.; Branch, J.W. Detection of Motorcycles in Urban Traffic Using Video Analysis: A Review. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 6115–6130. [CrossRef]
26. Gepperth, A.; Dittes, B.; Garcia-Ortiz, M. The contribution of context information: A case study of object recognition in an intelligent car. *Neurocomputing* **2012**, *94*, 77–86. [CrossRef]
27. Bileschi, S.M. StreetScenes: Towards Scene Understanding in Still Images. Ph.D. Thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Cambridge, MA, USA, 2006. Available online: https://dspace.mit.edu/handle/1721.1/37896 (accessed on 30 March 2024).
28. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets Robotics: The KITTI Dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
29. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. Available online: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/ (accessed on 30 March 2024).
30. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. Available online: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/ (accessed on 30 March 2024).
31. Griffin, G.; Holub, A.; Perona, P. CaltechDATA: Caltech 256. Available online: https://data.caltech.edu/records/nyy15-4j048 (accessed on 30 March 2024).

32. Thai, N.D.; Le, T.S.; Thoai, N.; Hamamoto, K. Learning bag of visual words for motorbike detection. In Proceedings of the 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, 10–12 December 2014; pp. 1045–1050.
33. Sutikno, S.; Waspada, I.; Bahtiar, N.; Sasongko, P.S. Classification of motorcyclists not wear helmet on digital image with backpropagation neural network. *Telkomnika Telecommun. Comput. Electron. Control.* **2016**, *14*, 1128–1133. [CrossRef]
34. Messelodi, S.; Modena, C.M.; Cattoni, G. Vision-based bicycle/motorcycle classification. *Pattern Recognit. Lett.* **2007**, *28*, 1719–1726. [CrossRef]
35. Duan, B.; Liu, W.; Fu, P.; Yang, C.; Wen, X.; Yuan, H. Real-time on-road vehicle and motorcycle detection using a single camera. In Proceedings of the 2009 IEEE International Conference on Industrial Technology, Churchill, VIC, Australia, 10–13 February 2009; pp. 1–6.
36. Nong, M.A.M.; Osman, R.; Yusof, J.M.; Sidek, R.M. Motorcycle image tracking and edge detections based on Simulink software. In Proceedings of the 2016 6th International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia, 15–17 August 2016; pp. 1–4.
37. Rashidan, M.; Mustafah, Y.; Shafie, A.; Zainuddin, N.; Aziz, N.; Azman, A. Moving object detection and classification using Neuro-Fuzzy approach. *Int. J. Multimed. Ubiquitous Eng.* **2016**, *11*, 253–266. [CrossRef]
38. Espinosa, J.E.; Velastin, S.A.; Branch, J.W. MB7500 Data Set. Available online: http://videodatasets.org/UrbanMotorbike (accessed on 30 March 2024).