Hindawi
Journal of Sensors
Volume 2022, Article ID 6335118, 13 pages
https://doi.org/10.1155/2022/6335118

Research Article

Performance Evaluation of Deep Learning Algorithm Using High-End Media Processing Board in Real-Time Environment

Muhammad Asif[1], **Tabarka Rajab**[1], **Samreen Hussain**[2], **Munaf Rashid**[1], **Sarwar Wasi**[1], **Areeb Ahmed**[1] and **Kehkashan Kanwal**[3]

[1]*Data Acquisition, Processing, And Predictive Analytics Lab, NCBC, Ziauddin University, Karachi, Pakistan*
[2]*Aror University of Art, Architecture, Design and Heritage, Sukkur 65400, Pakistan*
[3]*Department of Electrical Engineering, Faculty of Engineering, Science, Technology and Management (ZUFESTM), Ziauddin University, Pakistan*

Correspondence should be addressed to Sarwar Wasi; sarwerwasi@gmail.com

Received 14 June 2022; Accepted 30 August 2022; Published 7 December 2022

Academic Editor: Sangsoon Lim

Image processing-based artificial intelligence algorithm is a critical task, and the implementation requires a careful examination for the selection of the algorithm and the processing unit. With the advancement of technology, researchers have developed many algorithms to achieve high accuracy at minimum processing requirements. On the other hand, cost-effective high-end graphical processing units (GPUs) are now available to handle complex processing tasks. However, the optimum configurations of the various deep learning algorithms implemented on GPUs are yet to be investigated. In this proposed work, we have tested a Convolution Neural Network (CNN) based on You Only Look Once (YOLO) variants on NVIDIA Jetson Xavier to identify compatibility between the GPU and the YOLO models. Furthermore, the performance of the YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv5s models is evaluated during the training using our PowerEdge Dell R740 Server. We have successfully demonstrated that YOLOV5s is a good benchmark for object detection, classification, and traffic congestion using the Jetson Xavier GPU board. The YOLOv5s achieved an average precision of 95.9% among all YOLO variants and the highest success rate achieved is 98.89.

1. Introduction

In recent years, technology has constantly been evolving with the spread of artificial intelligence techniques such as deep learning. Various machine learning algorithms have been developed to solve one of the biggest challenges in computer vision, namely, object detection and identification [1]. Object detection is a problem of identification, localization, and classification of single or multiple objects in an image [2]. It is well established that deep learning algorithms have shown superior results to conventional techniques. There are two main categories for object detection, identification, and tracking. The first is based on a single-stage neural network with convolutional architecture [3] that generates a fixed number of predictions on the grid, such as SSD [4], YOLO [5], and M2Det [6]. The latter is based on two or more stage networks that take advantage to find regions of interest that have a high probability of containing an object and second or higher networks to get the classification score and spatial offsets, such as FPN [7], YOLOv5 [8], and faster R-CNN [9]. Object detection techniques have been successfully used in many real-time applications ranging from autonomous driving [10] to robotics and machine vision [11], video surveillance [12] to traffic monitoring [13], and medical imaging [14] to the diagnostic system [15].

Despite the unimaginable breakthroughs in machine learning and deep learning [16], there is plenty of room for improvement. In the present era, object detection, identification, and tracking depend on an efficient algorithm and an embedded platform running the computationally expensive

Table 1: A brief overview of various deep learning algorithms implemented over GPUs or conventional computing devices for a variety of object detection purposes.

Deep learning algorithms implemented over conventional computing devices			
Author and publication year	Hardware	Algorithm	Aim
Artamonov et al. [22]	NVIDIA Jetson/Tegra	YOLO- CNN	Traffic sign recognition
Barba-Guaman et al. [26]	Jetson Nano	SSD-Mobilenet V1 and V2 (single shot detector), SSD-inception V2, and PedNet, multiped	Vehicle and pedestrian detection
Komasilovs et al. [12]	Intel i5, 16 GB RAM	SSD Mobilenet V1 model	Traffic sign recognition
Castellano et al. [27]	NVIDIA GeForce MX110 (2 GB), RPi3 NVIDIA Jetson TX2	Lightweight FCN	Crowd detection
Zhao et al. [6]	LiDAR(light detection and ranging) sensors, NVIDIA GTX 1080i GPU	Spiking convolutional neural network in YOLOv2	Vehicle and pedestrian detection and minimize power consumption of LiDAR
Avramović et al. [25]	GeForce GTX 1080 Ti	YOLO variants	Traffic sign recognition
Khazukov et al. [24]	GPU: GeForce RTX 2080 TI, CPU: i9 9900k, RAM: 64 GB	YOLOv3	Speed detection and classification
Komasilovs et al. [12]	Cameras, CPU Intel i5, 16 GB RAM	SSD Mobilenet V1 model	Object detection and tracking
Blair and Robertson [21]	FPGA(field programmable gate array), GPU, and CPU	HOG, MoG (Histogram of Oriented Gradient and Mixture of Gaussian)	Object tracking/event detection

algorithm. The optimal selection of embedded platforms is critical for real-time applications [17].

Over the last few years, embedded hardware has intensified as platforms with graphical processing units (GPU) [18]. Embedded platforms based on GPUs provide high performance and low power consumption and perform functions in parallel. In addition, the compatibility of the NVIDIA GPU-based embedded system with the JetPack SDK [19] and other Open-Source Computer Vision Library (OpenCV) provides good advantages as they have libraries for deep computer vision learning and accelerated computing. However, the performance of an embedded system based on GPU depends on various parameters such as GPU and memory usage, temperature, and inference time [20]. Furthermore, most reported works use offline or batch mode where historical or recorded data sets are analyzed. This article examines the performance of NVIDIA Jetson Xavier using deep learning algorithms in real-time environments. The performance of all standard YOLO variants for YOLOv1, YOLOv2, YOLOv3, YOLOv4, and YOLOv5 are tested and evaluated in real-time on NVIDIA Jetson Xavier. The main contributions that have been made through this research work are as follows:

(1) The performance of YOLO variants with improved CNN algorithms is evaluated in real-time

(2) The performance parameters of NVIDIA Jetson Xavier AGX, such as memory, temperature, and interference time, are also measured and evaluated with the real-time implementation of YOLO variants

(3) The GPU processing board NVIDIA Jetson Xavier was evaluated to analyze the real-time road traffic performance using real-time traffic data

Furthermore, the remaining paper is structured as follows. Section 2 discusses the literature review on recent research based on CNN-based object detection algorithms. Section 3 presents the methodology of the research papers, and analytical results are then provided in Section 4. Finally, Section 5 states the conclusions.

2. Literature Review

Table 1 summarizes the literature on various deep learning algorithms applied to GPUs over the last few years. In [21], Blair and Robertson used the Histogram of Oriented Gradients (HOG) and Mixture of Gaussian (MoG) for object and event detection on-field programming gate array (FPGA), central processing unit (CPU), and GPU. They concluded that the detectors using GPUs process faster and consume more power. However, the setups that perform processing on FPGA have relatively less power consumption with less accuracy. In [22], Artamonov et al. implemented YOLO on mobile graphic processors such as NVIDIA Jetson for traffic sign recognition. Komasilovs et al. developed a vehicle detection and tracking system using an outdoor surveillance camera. The pretrained SSD MobileNet V1 model

Table 2: Compares various Jetson embedded boards in terms of performance features taken from [17] adapted from [39].

Characteristics	Comparison of different NVIDIA Jetson boards				
	Jetson Nano	Jetson TX1	Jetson TX2	Jetson Xavier NX	Jetson AGX Xavier
CPU (central processing unit)	Quad-core ARM A57, 1.43 GHz	Quad-core A57 processor	Quad-core ARM A57 BD Dual-core Denver 2 64-bit CPU	6-core NVIDIA Carmel ARM 64-bit CPU 6 MB L2 +4 MB L3	8-Core ARM 64-bit CPU, 8 MB L2 +4 MB L3
GPU	128 core NVIDIA Maxwell	NVIDIA Maxwell GPU with 256 CUDA cores	N Pascal with 256 NVIDIA CUDA cores	NVIDIA Volta, 384 CUDA cores, and 48 tensor cores	512-CUDA cores and 64 tensor cores
DL accelerator (deep learning accelerator)	None	None	None	2× NVDLA engines	(2×) NVDLA engine
Memory	2 GB 64-bit LPDDR4	4 GB LPDDR4 memory	8 GB 128-bit LPDDR4	8 GB 128-bit LPDDR4× 59.7 GB/s	16 GB 256-bit LPDDR4x
Storage	MicroSD	16 GB eMMC 5.1	32 GB eMMC 5.1	MicroSD	32 GB eMMC 5.1
Video encoder	4Kp30-4× 720p30 (H.264/ H.265)	4 K at a rate of 30	2 × 4 K at a rate of 30 (HEVC)	2 × 4 K60 – 4 × 4 K30, 20× 1080p30 (H.264)	8 × 4 K at a rate of 60 (HEVC)
Video decoder	4Kp60-2×18× 720p30 (H.264/ H.265)	4 K rate of 30	2 × 4 K at 30, 12-bit support	2 × 8 K30-6× 4 K60 1080p 30 (H.265) 22× 1080p30 (H.264)	2 × 4 K at a rate of 30 12-bit support
Cost	$99	$520	$599	$399	$699

is used for the fine training vehicle detection model. The real-time tracking was done using a CPU (Intel i5, 16 GB RAM) and achieved an average of 92% vehicle detection and tracking accuracy. They concluded that the deep learning detection model is viable only when executed on better GPU-equipped hardware.

Zhou et al. [23] used automotive light detection and ranging (LiDAR) sensors and NVIDIA GTX 1080i GPU to implement a spiking convolutional neural network in YOLOv2 for real-time object detection for autonomous cars. The proposed networks were compared with various other frameworks. They concluded better performance in terms of average precision than other typical models reported in the literature.

Khazukov et al. [24] used the YOLOv3 on a CPU equipped with a GPU to detect vehicles and monitor traffic parameters. They use the YOLOv3 neural network architecture and Simple Online and Real-time Tracking (SORT) open-source tracker. They achieved almost 90% accuracy of vehicle count in day and night images. In [25], Avramović et al. used different variants of YOLO implemented on GeForce GTX 1080 to identify real-time performance for automotive applications, including driving assistance, detecting-road objects, autonomous vehicles, and automatic traffic sign inventory maintenance.

Barba-Guaman et al. [26] used the Jetson Nano and implemented various algorithms to detect vehicles and pedestrians' luggage, namely, single short detection (SSD), PedNet, multiped, mobile net V1 and V2, and SSD-inception V2. They used different datasets for the identification of vehicles and pedestrians. The maximum accuracy for vehicle detection was 84.01% for SSD-Mobilenet V1 and SSD-inception V2. In the case of pedestrian detection, the maximum obtained accuracy was 90.23% with the PedNet framework. They also found models that consume less time in their performance which were SSD-mobilenet-V2, SSD-mobilenet-V1, and SSD inception-V2.

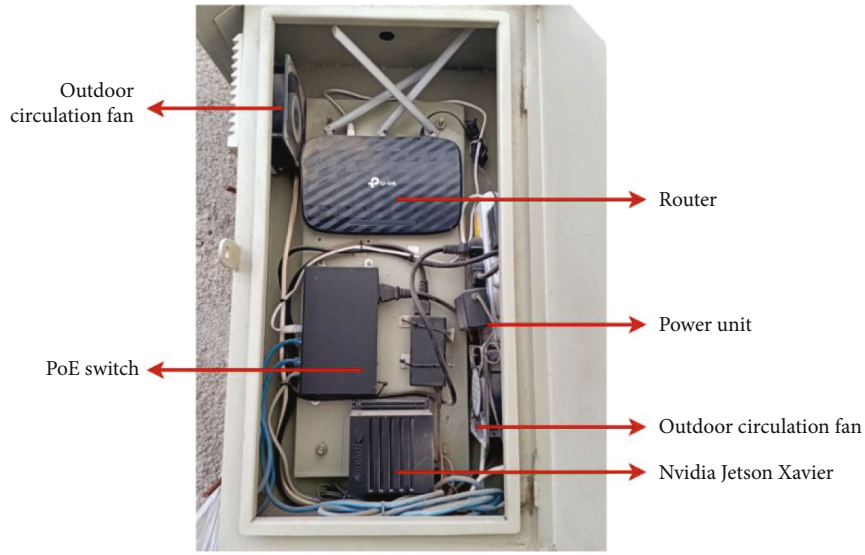
Table 3: Data collection description.

Description of data collection		
Dataset	Size	Object
COCO dataset [41]	330 K	80
Our dataset	10 K	5

Castellano et al. [27] used embedded hardware platforms to detect human crowds for aerial images using CNN. The training was run offline on the VisDrone dataset [28] using an Intel Ci5 system with 8 GB RAM and NVIDIA 2 GB GeForce MX110 GPU, running Windows 10 Operating system. The trained networks were deployed on two computational hardware platforms, Raspberry Pi 3, and NVIDIA Jetson TX2, and TX2 outperformed Raspberry Pi 3 regarding detection accuracy and processing speed for all implemented models.

Kim et al. [29] used a multistage convolutional neural network (MSCNN) and variants of YOLOv3 to improve vehicle detection on conventional Intel i5 CPU. The proposed MSCNN and YOLOv3 apply to three datasets: KITTI VD [30], AUTTI [31], and crowd AI [32]. The algorithms were trained using the Pytorch package [33] in Python and the GTX TITAN X GPU. Tests were performed on Intel CPU i5-4670 without a dedicated GPU; however, the

(a) Location of the cameras for data collection



(b) Standalone system

Figure 1: Locations of camera and hardware prototype of the proposed system. (a) Top picture shows the dataset collection performed from different Ziauddin university and hospital locations by placing the cameras. (b) Bottom image shows the processing unit being used in performing the proposed method.

feasibility of real-time embedded implementation of MSCNN and YOLOv3 was not discussed.

It is important to note that the previous works are based on either offline or batch mode, where historical or recorded datasets are used for object detection, identification, and tracking. They have been using computationally less expensive algorithms, which provide compromised accuracy in terms of detection and tracking. Other methods discussed in the literature can often be considered expensive, given the real-time requirements, implementation on embedded platforms, and application's computational limits.

2.1. Object Detection Algorithms. There are various object recognition and detection algorithms, such as YOLO [34]. However, YOLO (You Only Look Once) gained significant importance in the computer vision community due to its real-time and accurate object detection in wide applications [35]. YOLOv2 was released in 2017 with several iterative improvements in the layers, including batch norm. A higher version added an object score presented in the bounding box YOLOv2 was released in 2017 with several iterative improvements in the layers, including batch norm, higher resolution, and an adequately defined anchor box.

YOLOv3 was released in 2018, and the improvements applied to this version added an object score presented in the bounding box prediction and improved backbone network layers. These predictions are at three stages of granularity by improving the performance of smaller objects. YOLOv2 and YOLOv3 have improved and higher mAP; FPS than the Faster R-CNN and SSD, whereas Girshick first published RCNN [36] and faster R-CNN [37] in 2014 and 2015.

YOLOv4 was released in 2020 from the literature and has been more accurate than the YOLOv3 algorithm. However, the accuracy of YOLOv4 has been compared with YOLOv5, which is still open to question as some researchers have been claiming that YOLOv4 is the more accurate while others are claiming that YOLOv5 is more accurate. Jocher et al. released YOLOv5 in 2020, right a few days after the release of the YOLOv4 algorithm, with enhanced improvements. The reported results show that all attributes have different datasets and improved hyperparameters, since none of the related works uses the real-time live streaming processing on

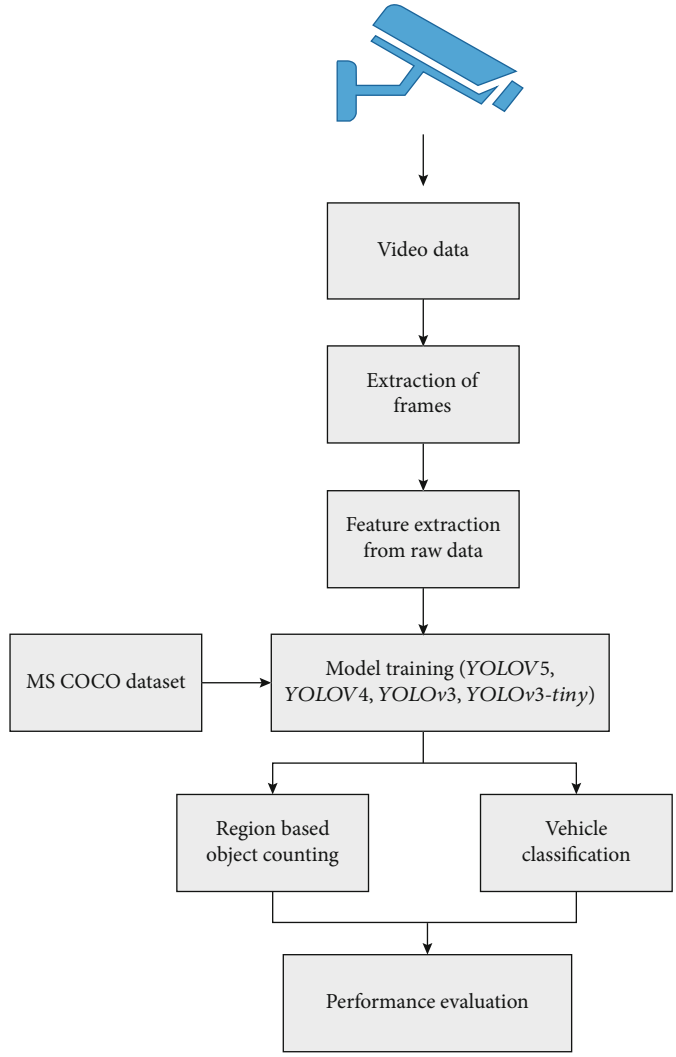


Figure 2: The system workflow: the video camera records stream, broken down into frames, and extracted features. The algorithm trained on the COCO dataset then performs region-based detection, counting, and classification. Finally, the performance of the system is evaluated.

NVIDIA Jetson AGX Xavier compared with different YOLO variants with the specific criteria in this research.

2.2. Mobile Computation Platforms. GPU is characterized by excellent memory bandwidth and computation power [38]. With the same number of transistors available, the GPU achieves higher arithmetic intensity due to graphic computation's parallel nature. In addition, the GPUs are both inexpensive and readily available. These features make the GPU an excellent choice for implementing deep learning frameworks. With the advent of developer kits like NVIDIA Jetson developer kits, real-time portable applications have been made possible. Table 2 presents NVIDIA Jetson developer boards available as a computation platform.

3. Methodology

This section proposes detecting and counting the vehicles for traffic congestion monitoring using hardware acceleration. In the present scenarios, different hardware accelerators can solve complex problems through their capabilities. This research includes the NVIDIA developer kit from the Jetson family (Jetson Xavier AGX) used for the application because of its high computation performance, as mentioned in Table 2, and power consumption efficiency in utilizing a standalone system. The Xavier AGX hardware accelerator selection depends on the data characteristics such as size, quantity, and application. This information provides assistance in selecting the right combination based on the data Properties in [40].

3.1. Data Selection. The initial step in selecting and implementing the deep learning-based YOLO-variant algorithms is to select the data. The data process involves converting videos into images, where resolution must be considered to accomplish good quality images to train the model and determine the size of the input of the algorithms. The data quantity should be observed because if the data is too small

Table 4: Comparison of various convolutional neural network (CNN) frameworks in terms of map, FPS, and layers.

Convolutional neural network (CNN) frameworks Comparision				
CNN variants	Author	mAP(mean average precision)	FPS(frame per second)	Layers
YOLO	Redmon et al. [34]	63.4	45	26
YOLOv2	Redmon et al. [42]	48.1	42	32
YOLOv3	Redmon et al. [4]	51.5	20	106
YOLOv3-tiny	Adarsh et al. [43]	33.1	220	24
YOLOv4	Bochkovskiy et al. [44]	43.5	65	137
YOLOv5	Jocher et al. [45]	48.1	264	24
YOLOv5s	Jocher et al. [45]	36.8	455	17
		Speed up	Test time per image (s)	
R-CNN	Saeidi and Ahmadi [36]	1×	47	
Faster R-CNN	Ren et al. [37]	146×	0.32	

for the training set that will negatively impact the samples for each class. The COCO dataset [41] is mentioned in Table 3, from which 20,500 images have been taken of five classes of traffic vehicles. Our dataset contains one week of a traffic video sequence at various challenging levels, such as variations in time duration according to nighttime and daytime, with different angles on a high-resolution and augmented the dataset with a different rotation, zooming, and flip into top and bottom. The first step is to divide the dataset into 80% for training and 20% for testing. The first step is to consider the color samples from the combined dataset from the contained information for the feature extraction, using it for deep learning models.

3.2. Hardware Accelerator. DL methods rely on hardware accelerators, particularly those that fulfil data needs, and the application of the model must be chosen, necessitating assessment to find the best hardware for this processing. Deep neural networks' growth has raised the need for computational complexity and, as a result, their resource consumption, providing implementation issues for deep neural networks. As mentioned in Table 2, the NVIDIA Jetson Xavier AGX has been used, utilizing power consumption efficiency as a standalone system. However, the other components of the prototype hardware include an ethernet-supported Hikvision HD camera and a battery backup system for portability. The camera (model: DS-2CD4A85-IZH and resolution ultra HD 4 K resolution) [46] is communicated with NVIDIA Jetson Xavier AGX using Real-Time Streaming Protocol (RTSP) for live data analysis, as shown in Figure 1. Furthermore, Internet Protocols (IPs) assigned to the camera and NVIDIA media processor are of the same IP pole, 192.168.1.2 and 192.168.1.3, respectively.

The core part of the proposed AI-enabled object detection-based traffic monitoring system is developing and training several YOLO variants. Figure 2 represents the overall process followed. However, further explanation regarding the implementation is explained below sections.

3.3. Implementation of YOLO Variants. The You Only Looks Once (YOLO) deals with object detection, which takes an image as an example and predicts it by its bounding box coordinates. The YOLO algorithm locates each object and a corresponding class label using a bounding box and has an advantage in speed and performance compared to other deep learning algorithms. YOLO uses the convolutional neural network backbone, divided into three layers: input, hidden, and output. However, Table 4 mentions the total layers of individual YOLO models. The YOLO works well for multiple objects. Each object is associated with one grid cell, which helps overlap where one grid cell contains the center points of the two different objects known as anchor boxes. Each bounding box in the anchor box contains a certain height and width. Figure 3 illustrates a field test where the YOLO detects multiple objects in the image.

3.3.1. Loss Function.

$$\begin{aligned}\text{Loss} = {} & \lambda_{\text{coord}} \sum_{i=0}^{s2} \sum_{j=0}^{b} l_{ij}^{V} \left[\left(x_i - \hat{x}_i \right)^2 + \left(\left(y_i - \hat{y}_i \right)^2 \right] \right. \\ & + \lambda_{\text{coord}} \sum_{i=0}^{s2} \sum_{j=0}^{b} l_{ij}^{v} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{s2} \sum_{j=0}^{b} l_{ij}^{v} \left[\left(C_i - \hat{C}_i \right)^2 \right] + \lambda_{\text{novehicle}} \sum_{i=0}^{s2} \sum_{j=0}^{b} l_{ij}^{v} \left(C_i - \hat{C}_i \right)^2 \\ & + \sum_{j=0}^{s^2} l_{ij}^{v} \sum_{c \epsilon \text{class}}^{s^2} \left(V(c) - \hat{V}i(c) \right)^2. \end{aligned} \tag{1}$$

From equation (1), the following are:
S refers to the number of grids;
b total number of forecasting boxes in a cell;
x and y different for each cell, center coordinates;
w and h are dimensions of the prediction box;
C confidence of forecasting;
v vehicle detection assurance;
λ_{coord} position loss function weight;
$\lambda_{\text{novehicle}}$ classification loss function weight;

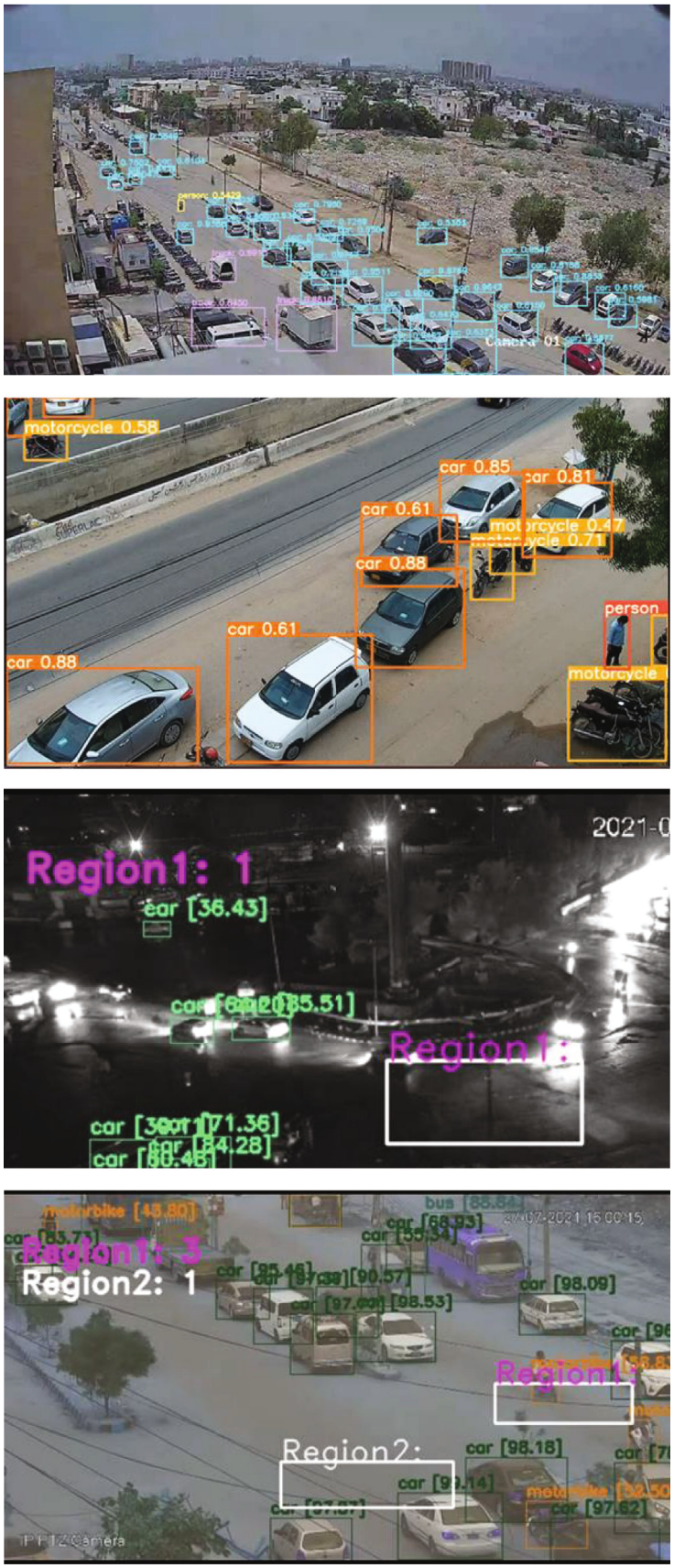


Figure 3: Detection results from the designed algorithms. Each object has a bounding box drawn over it, and a label is assigned by classification.

Figure 4: Image labeling using the labeling tool: labels are created, and each object is assigned one label.

l_{ij}^{v} vehicle object in the j^{th} prediction frame; in case of a target vehicle, its value is 1, otherwise, 0;

subsequent predicted value = $(x, {}^{y}, w, h, C, V^{x}, {}^{y}, w, h, C, V)$.

The general loss function calculates the sum of the total squared error for position predictions. According to the root-square value of predictive width and height for the box, the third and the fourth summing elements utilize the loss function for certainty. The fifth part adds to the equation and utilizes the loss function for the likelihood class. In the YOLO calculation, the Intersection over Union (IOU) loss function error and the loss function error classification are determined using multiclass crossentropy classification.

3.3.2. Training and Evaluation. The network has been trained using MS COCO (Common Objects in Context) dataset. The dataset has 80 classes of objects with annotations and labels. However, our dataset has combined MS COCO to improve the algorithm's performance. For dataset development, HD cameras were deployed at various locations (Dr Ziauddin Hospital North Nazimabad, Dr Ziauddin Hospital OPD North Nazimabad, and Ziauddin Engineering University) and converted the video stream into frames, and then those frames were saved into annotations of text file format by using the label Img tool as demonstrated in Figure 4. The total images taken were 5000 at this stage. The dataset has been augmented using the library of augmentor. The purpose was to train the algorithm to face real-time data challenges like noise, picture brightness variation, and frame tilting issues. The five classes of images were considered for training the network: truck, bus, car, bicycle, and motorbike. The Dell R740 Server combined with NVIDIA Tesla T4 GPU is utilized for training the models, as shown in Figure 2.

For evaluation, trained algorithms YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv5s on a high-definition live video stream of 20 fps have been tested. The research work is evaluated using YOLO models, and the performance matrices are given in. The presented work is performed using the NVIDIA Jetson AGX Xavier controller. The processing is evaluated based on the RAM utilization, inference time, temperature, and GPU utilization at a different resolution. The overall hardware and software packages required for system training and testing are shown in Table 3.

4. Results and Discussion

The performance of the YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv5s models is evaluated during the training, and the parameters are mentioned in Table 5. A 5000-image dataset is used for evaluating each algorithm. The models YOLOv3-tiny, YOLOv3, and YOLOv4 are quite similar, but the YOLOv5s are adaptive learners and have high precision and recall compared to other YOLO models. In Table 5, the first column represents the labels of the trained objects divided into five classes: car, truck, motorbike, bicycle, and bus. The second column represents the image size of images during the training, and the third column is for batch size for the model; the fourth and fifth represent recall and precision. Finally, the sixth and seventh column is average precision for the performance of the proposed model. The latest variant of the YOLO family, defined as YOLOv5s, has high precision, recall, and reduced weight size, which is the lightest weight characteristic compared with the other models.

$$\text{Precision}\,(P) = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2}$$

$$\text{Recall}\,(R) = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3}$$

$$\text{AP} = \sum_{i=0}^{i=n-1} [R(i) - R(i+1)] * Pi, \tag{4}$$

$$\text{mAP} = \sum_{i=1}^{N} \frac{APi}{N}. \tag{5}$$

In Equation (2), The true positive (TP) is used for correctly detecting any object that represents and exists in the frame conducted from the video. False-positive (FP) represents the invalid/incorrect detection (sometimes, the algorithm detects the incorrect objects in the frame while detecting an object). In Equation (3), a false negative (FN) represents the object the algorithm does not detect. The Intersection over Union (IoU) assesses the overlap region among the forecasting box and the actual item's ground truth bounding box in object detection. It can be categorized as correct or wrong by comparing the IoU to a specified threshold using IoU. Equations (4) and (5) are used for average computing precision (AP) and mean average precision. AP has been used to show the precision and recall curve into a numeric value representing the

Table 5: Environments for research implementation.

Implementation of hardware and software in implementing the algorithms			
Hardware environment		Software environment	
Memory in GB	256	Operating system	18.0 Ubuntu
Processing unit (CPU)	Intel Xeon Silver 4214 CPU 2.20GHz ×48	Environment	Python 3.6
GPU	NVIDIA Jetson Xavier NVIDIA Tesla T4		Pytorch framework

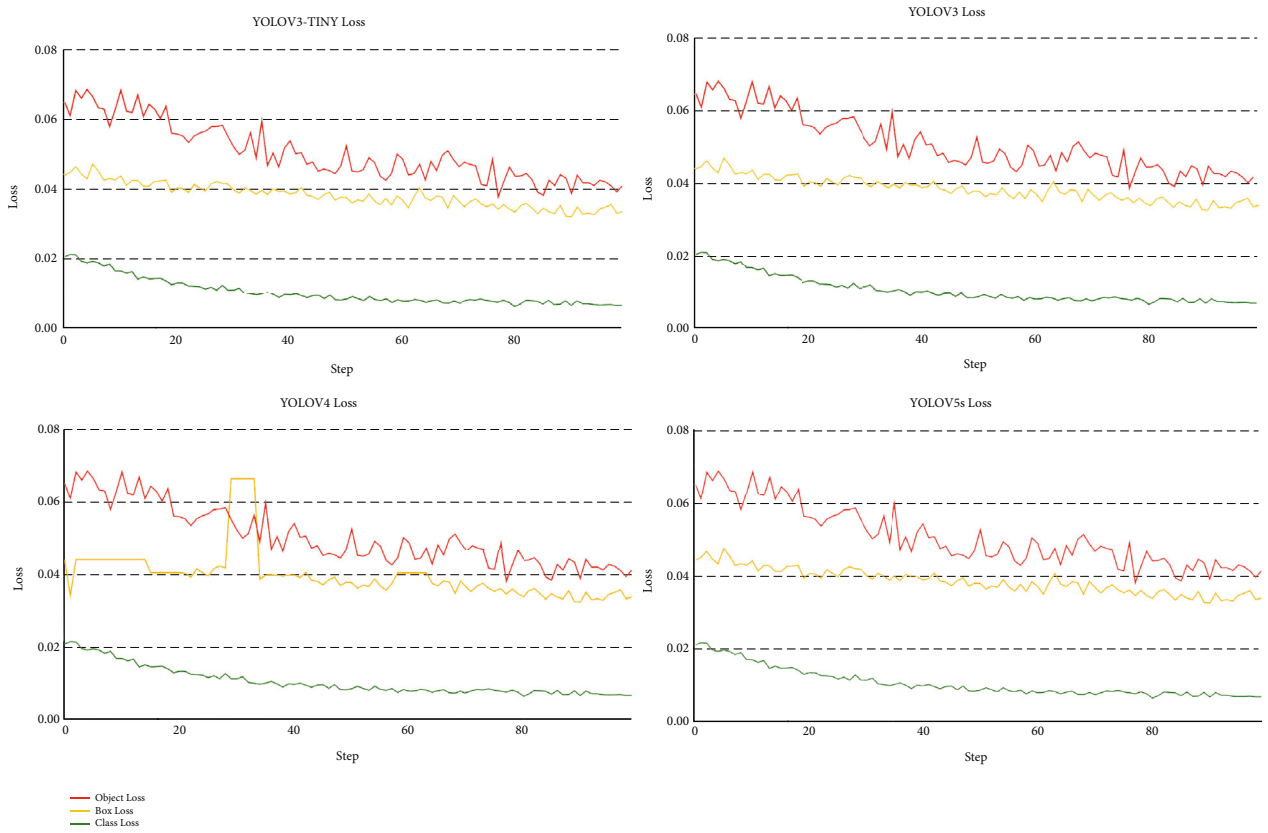


Figure 5: Results of loss on YOLOv5, YOLOv4, YOLOv3, and YOLOv3-tiny algorithms. Each loss declines over iterations.

overall precision average, where n is defined as the number of thresholds. The AP is the weighted sum of precision at each threshold, corresponding to the increase in recall. However, mAP is calculated with a value between 0 and 1, indicating how much the anticipated and ground truth bounding boxes overlap. Because each value of the IoU threshold yields a precise average accuracy (AP) measure, this value must be specified.

The trends in Figure 5 evaluate the loss graph of YOLOv5s, YOLOv4, YOLOv3, and YOLOv3-tiny for class, object, and box. The horizontal axis shows the graph iterations, and the vertical axis represents the loss amplitude. However, the trend shows that the overall performance of the individual model is similar, and the loss trend decreases at a similar rate with each number of iterations (Table 6) exhibits classification performance by presenting the YOLOv5s, YOLOv4, YOLOv3, and YOLOv3-tiny model training results. Table 7 represents the success rate for classifying each object on the scene and instances of misclassification. The highest success rate of 98.89% has been obtained for cars, whereas the algorithm has correctly identified the bus. The lowest success rate obtained is 89.88% for bicycles. Overall, the misclassifications are within tolerable limits. Figure 3 below illustrates the objects detected by the deep learning models by making bounding boxes around the detected image.

The performance evaluation of the models in terms of RAM utilization, inference time, GPU utilization, and the temperature of the on-shelf controller is presented in Figures 6–9, respectively. For YOLOv3-tiny (in Figure 6), GPU temperature remains on the lower side, whereas for YOLOv5s (in red in Figure 6), the higher temperature corresponded to higher image resolutions of $2304 * 1296$ and

Table 6: Training results of the YOLOv5s, YOLOv4, YOLOv3, and YOLOv3-tiny Model.

Model	Labels	YOLO training parameters Image size	Batch size	Size	Recall	Precision	mAP
YOLOv5s	5	640 x 640	16	14 MB	0.98	0.987	0.959
YOLOv4				245 MB	0.90	0.899	0.886
YOLOv3				236 MB	0.91	0.909	0.897
YOLOv3-tiny				33.7 MB	0.949	0.932	0.90

Table 7: Classification performance analysis for the proposed system.

Objects	Success rate	% misclassifications Car	Motorbike	Bus	Truck	Bicycle
Car	98.89%	—	—	—	1	—
Motorbike	95.84%	—	—	—	—	2
Bus	93.72%	—	—	—	—	—
Truck	93.72%	1	—	—	—	—
Bicycle	89.88%	—	2	—	—	—

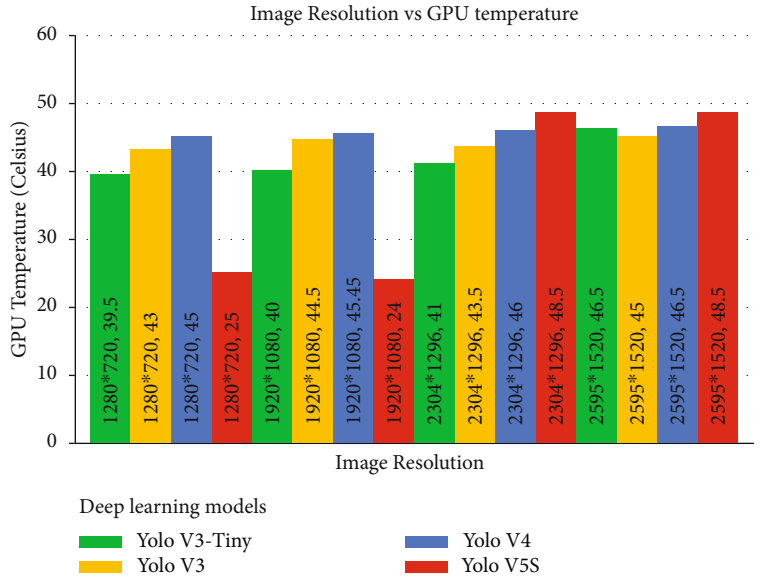


Figure 6: Effect of image resolution on GPU temperature for the deep learning models used for the research as these model's inference time depends on the resolution. If resolution increases, the GPU temperatures increase as well.

2595 ∗ 1520. YOLOv3 and YOLOv4 (in Figure 6) also had considerably high temperatures for all resolutions. In Figure 6, YOLOv5s at lower resolution seem good in keeping the GPU temperature low. However, the temperature rapidly increases at higher resolution, which shows that the v5s utilize more GPU. The inference time (in seconds) is one of the crucial factors in live streaming tasks. However, it is highly dependent on detection accuracy. As illustrated in Figure 7 below, the inference time increased as the resolution of images was generally in YOLOv3(in Figure 7), obtaining the highest inference of 3.13 with 2596 ∗ 1520 resolution. However, YOLOv3-tiny and YOLOv4 maintain a lower inference time than others, with the lowest value of 0.46 and 0.6 seconds, respectively. On the other hand, YOLOv5s take 1.37 and 2.579 seconds at resolutions 1280 ∗ 720 and 2596 ∗ 1520, respectively.

The histogram in Figure 8 illustrates the relation between the image resolution and GPU utilization. YOLOv5S (in Figure 8) has been shown to use more GPU than other models for all image resolutions, whereas YOLOv3-tiny (in Figure 8) used the lowest GPU for all image resolutions. Therefore, YOLOv3 tiny seems to be the most reasonable option for implementing the algorithm on the Jetson Xavier board if the most significant concern is

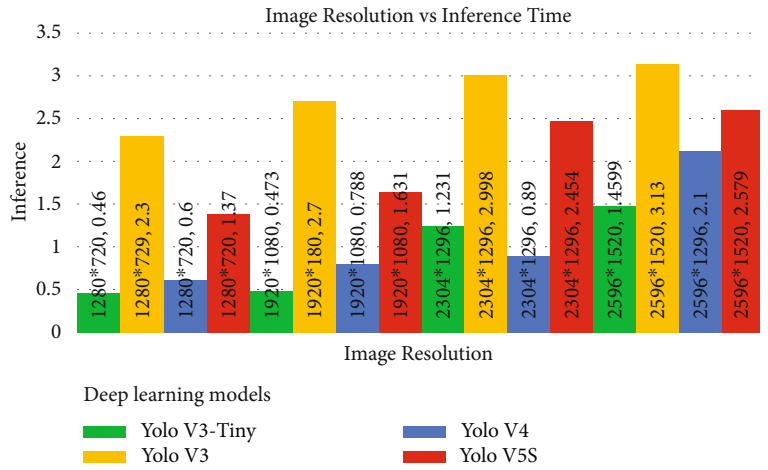


Figure 7: Effect of image resolution on inference for the deep learning models used for the research.

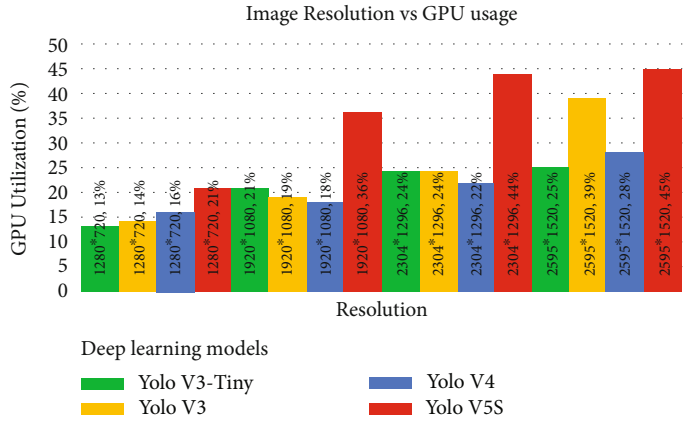


Figure 8: Effect of image resolution on GPU utilization by the employed deep learning models.

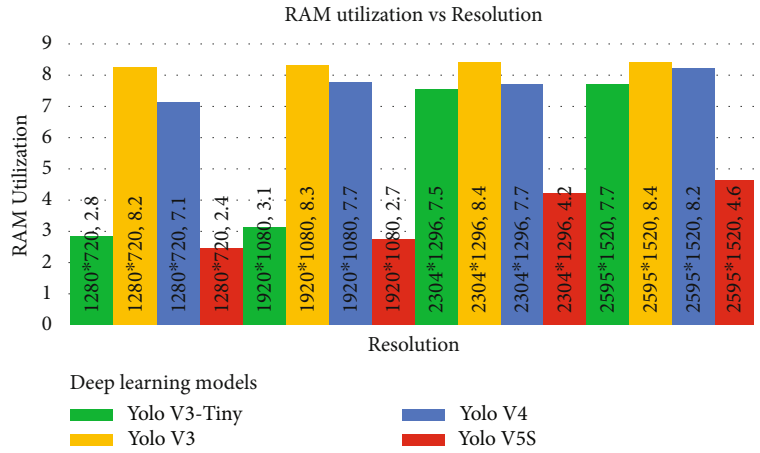


Figure 9: Effect of image resolution on utilization of RAM for each employed deep learning models.

keeping the GPU utilization in check. Finally, the lowest RAM utilization has been achieved by YOLOv5s, as shown in Figure 9 as a general trend, and RAM utilization exhibits an ascending trend as image resolution is increased. For example, YOLOv3-tiny and YOLOv5s utilize 2.8 and 2.4 GB of RAM, respectively, out of 16 GB at lower resolution. However, YOLOv3 uses 8.2 GB, and YOLOv4 uses 7.1 GB of RAM at a resolution of $1280 * 720$.

5. Conclusion and Future Work

The main scientific contribution of the project is developing a standalone system using Jetson Xavier AGX to perform traffic surveillance and monitoring. In this research paper, Jetson Xavier AGX is an excellent choice for implementing complex CNN-based (YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv5s) models with exceptional performance. Furthermore, improving deep learning libraries on NVIDIA platforms can result in even better results. The proposed system has been tested day and night, showing a success rate of 98.895. Traffic monitoring and management are one of the biggest challenges for third-world countries like Pakistan. By implementing the presented systems to detect and count vehicles, traffic problems can be minimized, such as false parking detection, traffic management using traffic controlling, and congestion detection. Finally, the work has successfully demonstrated that the powerful computational abilities of Jetson Xavier AGX can be exploited for object detection in live video streams.

Data Availability

The data is available and can be accessed on request, and contact the authors for further assistance at sarwerwasi@gmail.com and tabarkaa.r@gmail.com.

Conflicts of Interest

There are no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the whole research lab's help and support provided by Data Acquisition, Processing, and Predictive Analytics Lab, National Center in Big Data and Cloud Computing, Ziauddin University, Karachi, Pakistan.

References

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: a survey," 2019, https://arxiv.org/abs/1905.05055.

[2] Y. Amit, *2D object detection and recognition: Models, algorithms, and networks*, MIT Press, 2002.

[3] S. K. Raza, T. Rajab, S. A. Ahmed, and M. Khurram, "Visual landmarks recognition of urban structures using convolutional neural network," in *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–9, Sukkur, Pakistan, 2020.

[4] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[5] Y. Wageeh, H. E. D. Mohamed, A. Fadl et al., "YOLO fish detection with Euclidean tracking in fish farms," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 5–12, 2021.

[6] Q. Zhao, T. Sheng, Y. Wang et al., "M2det: A single-shot object detector based on multi-level feature pyramid network," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9259–9266, 2019.

[7] S. Shin, H. Han, and S. H. Lee, "Improved YOLOv3 with duplex FPN for object detection based on deep learning," *The International Journal of Electrical Engineering & Education*, 2021.

[8] Z. Zheng, J. Zhao, and Y. Li, "Research on detecting bearing-cover defects based on improved YOLOv3," *IEEE Access*, vol. 9, pp. 10304–10315, 2021.

[9] Y. Wang, S. M. A. Bashir, M. Khan et al., "Remote sensing image super-resolution and object detection: benchmark and state of the art," *Expert Systems with Applications*, vol. 197, article 116793, 2022.

[10] J. Levinson, J. Askeland, J. Becker et al., "Towards fully autonomous driving: systems and algorithms," in *2011 IEEE intelligent vehicles symposium (IV)*, pp. 163–168, Baden-Baden, Germany, 2011.

[11] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[12] V. Komasilovs, A. Zacepins, A. Kviesis, and C. Estevez, "Traffic monitoring using an object detection framework with limited dataset," in *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2019)*, pp. 291–296, Heraklion, Crete, Greece, 2019.

[13] E. Dizon and B. Pranggono, "Smart streetlights in Smart City: a case study of Sheffield," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, pp. 1–16, 2021.

[14] T. G. Nick and K. M. Campbell, "Logistic regression," *O Biologico*, vol. 404, pp. 273–301, 2007.

[15] R. Kundu, R. Das, Z. W. Geem, G. T. Han, and R. Sarkar, "Pneumonia detection in chest X-ray images using an ensemble of deep learning models," *PLoS One*, vol. 16, no. 9, article e0256630, 2021.

[16] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2503–2510, Seattle, WA, USA, 2018.

[17] "Embedded systems developer kits & modules from NVIDIA Jetson," 2021, https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/.

[18] U. Dastgeer and C. Kessler, "Smart containers and skeleton programming for GPU-based systems," *International Journal of Parallel Programming*, vol. 44, no. 3, pp. 506–530, 2016.

[19] "JetPack SDK | NVIDIA Developer," 2022, https://developer.nvidia.com/embedded/jetpack.

[20] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: a survey," *Integration*, vol. 69, pp. 309–320, 2019.

[21] C. G. Blair and N. M. Robertson, "Video anomaly detection in real time on a power-aware heterogeneous platform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, pp. 2109–2122, 2016.

[22] N. S. Artamonov and P. Y. Yakimov, "Towards real-time traffic sign recognition via YOLO on a Mobile GPU," *Journal of Physics: Conference Series*, vol. 1096, no. 1, article 012086, 2018.

[23] S. Zhou, Y. Chen, X. Li, and A. Sanyal, "Deep SCNN-based real-time object detection for self-driving vehicles using LiDAR temporal data," *IEEE Access*, vol. 8, pp. 76903–76912, 2020.

[24] K. Khazukov, V. Shepelev, T. Karpeta et al., "Real-time monitoring of traffic parameters," *Journal of Big Data*, vol. 7, no. 1, 2020.

[25] A. Avramović, D. Sluga, D. Tabernik, D. Skočaj, V. Stojnić, and N. Ilc, "Neural-network-based traffic sign detection and recognition in high-definition images using region focusing and parallelization," *IEEE Access*, vol. 8, pp. 189855–189868, 2020.

[26] L. Barba-Guaman, J. E. Naranjo, and A. Ortiz, "Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU," *Electronics*, vol. 9, no. 4, 2020.

[27] G. Castellano, C. Castiello, C. Mencar, and G. Vessio, "Crowd detection in aerial images using spatial graphs and fully-convolutional neural networks," *IEEE Access*, vol. 8, pp. 64534–64544, 2020.

[28] D. Du, P. Zhu, L. Wen et al., "VisDrone-DET2019: the vision meets drone object detection in image challenge results," August 2022, http://www.aiskyeye.com/.

[29] J. Kim, S. Hong, and E. Kim, "Novel on-road vehicle detection system using multi-stage convolutional neural network," *IEEE Access*, vol. 9, pp. 94371–94385, 2021.

[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361, Providence, RI, USA, 2012.

[31] H. Caesar, V. Bankiti, A. H. Lang et al., "Nuscenes: a multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11618–11628, Seattle, WA, USA, 2020.

[32] S. P. Mohanty, J. Czakon, K. A. Kaczmarek et al., "Deep learning for understanding satellite imagery: an experimental survey," *Frontiers in Artificial Intelligence*, vol. 3, p. 85, 2020.

[33] A. Paszke, S. Gross, F. Massa et al., "PyTorch: an imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[35] J. Wu, B. Peng, Z. Huang, and J. Xie, "Research on computer vision-based object detection and classification," in *International Conference on Computer and Computing Technologies in Agriculture*, pp. 183–188, Berlin, Heidelberg, 2012.

[36] M. Saeidi and A. Ahmadi, "High-performance and deep pedestrian detection based on estimation of different parts," *The Journal of Supercomputing*, vol. 77, no. 2, pp. 2033–2068, 2021.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[38] J. D. Owens, D. Luebke, N. Govindaraju et al., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[39] S. Hossain and D. J. Lee, "Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with gpu-based embedded devices," *Sensors*, vol. 19, no. 15, p. 3371, 2019.

[40] M. Lopez-Montiel, U. Orozco-Rosas, M. Sanchez-Adame, K. Picos, and O. H. M. Ross, "Evaluation method of deep learning-based embedded systems for traffic sign detection," *IEEE Access*, vol. 9, pp. 101217–101238, 2021.

[41] T. Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," in *European conference on computer vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8693 of Lecture Notes in Computer Science, , pp. 740–755, Springer, Cham, 2014.

[42] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, San Juan, PR, USA, 2016.

[43] P. Adarsh, P. Rathi, and M. Kumar, "YOLO v3-tiny: object detection and recognition using one stage improved model," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 687–694, Coimbatore, India, 2020.

[44] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: optimal speed and accuracy of object detection," 2020, https://arxiv.org/abs/2004.10934.

[45] G. Jocher, K. Nishimura, T. Mineeva, and R. Vilariño, *YOLOv5 (2020)*, GitHub repository, 2020.

[46] "Hikvision DS-2CD4A85-IZH IP camera Specifications | Hikvision IP cameras," 2022, https://www.sourcesecurity.com/hikvision-ds-2cd4a85-izh-ip-camera-technical-details.html.