JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 11 Issue 02
2026

Research Article
https://doi.org/10.56979/1102/2026/1580

Efficient Deep Learning for Accurate Vehicle Classification in Urban Computing

Rana Muhammad Amir Latif[1], Atif Ikram[2,3*], Ali Salem Bin Sama[4], Lamia Hassan Rahamatalla[4], Najla Abdulaziz Almousa[4], and Marwan Abu-Zanona[4]

[1]Department of Computer Science, Superior University, Lahore, 54000, Pakistan.
[2]Department of Computer Science & IT, The University of Lahore, Lahore, 54000, Pakistan.
[3]Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu, Malaysia.
[4]Department of Management Information Systems, College of Business Administration, King Faisal University, Al-Ahsa, 31982, Saudi Arabia.
*Corresponding Author: Atif Ikram. Email: atif.ikram@cs.uol.edu.pk

Received: May 14, 2026 **Accepted:** August 18, 2026

Abstract: The rapid growth of urbanization has increased the need for reliable and automated vehicle classification systems to support intelligent transportation and urban computing applications. This study presents a comparative deep learning framework for classifying vehicle images into four categories: bicycle, bike, car, and truck. Seven widely used convolutional neural network (CNN) architectures VGG-16, VGG-19, Inception-V3, Inception-V4, ResNet-50, ResNet-101, and EfficientNet-B0 are evaluated using transfer learning and data augmentation. The dataset comprises RGB vehicle images collected from online image sources and organized into training and test sets. During training, image rescaling, shear transformation, zooming, and horizontal flipping are applied to increase training-data variability, while the test data are evaluated without geometric augmentation. Model performance is assessed using accuracy, loss, precision, recall, and F1-score. Among the evaluated architectures, EfficientNet-B0 achieves the highest reported classification accuracy of 97.06%, followed by ResNet-101 with 96.82%. EfficientNet-B0 also demonstrates strong class-level performance across all four vehicle categories, indicating its effectiveness for image-based vehicle classification. The comparative results provide insights into the suitability of different CNN architectures for vehicle classification within urban computing applications. However, the study evaluates image-level classification rather than object localization and does not explicitly assess challenging real-world conditions such as severe occlusion, nighttime illumination, or diverse camera viewpoints. Future work will therefore focus on larger and more diverse traffic datasets, computational-efficiency evaluation, real-world traffic-camera imagery, and deployment-oriented optimization for intelligent transportation systems.

Keywords: Vehicle Classification; Deep Learning; Convolutional Neural Networks; Urban Computing; Intelligent Transportation Systems

1. Introduction

The Human Vision System (HVS) is capable of performing some of these complex tasks with repeatability and accuracy. Some examples of the complex tasks that the Human Vision System (HVS) can perform with repeatability and accuracy are the detection, recognition, and identification of an object. The recent developments in the field of Computer Vision (CV) and Machine Learning (ML) and the availability of tools

such as faster GPUs, larger data sets, and better algorithms have enabled computers to accurately detect, identify, and categorize multiple objects in a video or image [1]. The main purpose of vehicle detection and categorization is the detection of vehicles in video or still images [2]. An important factor in traffic observation or traffic surveillance is the efficiency of locating vehicles. Using machine learning methods, Fig illustrates the number of cars spotted in traffic photos in Pakistan. To ensure public safety, autonomous vehicle detection systems must accurately identify moving objects in real time, such as automobiles, trucks, police cars, or motorcycles [3].

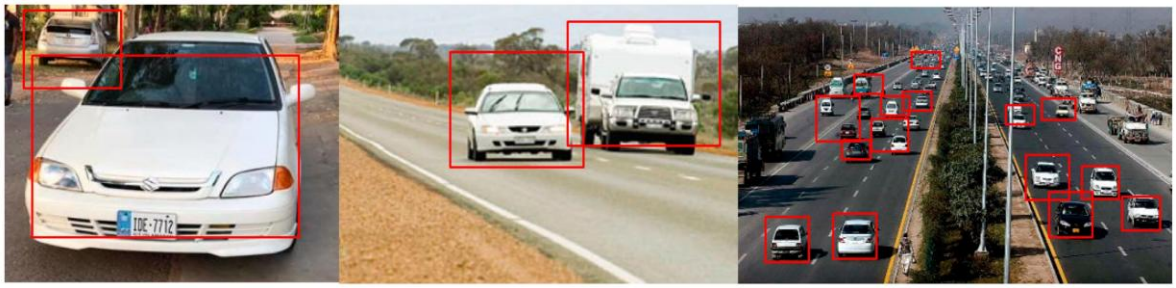


Figure 1. Examples of vehicle detection from traffic in Pakistan.

Due to advancements in DNNs, automated vehicle recognition has come a long way in the last several years. The applications where safety and reduced traffic are paramount include driver assistance systems and Autonomous Driving Systems (ADS) [4].

Vehicle localization is critical in developing intelligent and autonomous systems [5]. Such systems include self-directed driving, surveillance, object detection, and tracking. The capacity to detect only automobiles is the basis of the new high-tech innovation known as automatic driving [6]. Many crimes, including traffic violations, car accidents, and thefts, are captured on video by CCTV cameras in urban areas [7]. A reliable, accurate, and quick traffic detector is essential for real-time vehicle detection in traffic surveillance systems. There have been many developments in surveillance technology and traffic management systems [8]. While evaluating vehicle detectors, two key factors are the device's capacity to identify things in real-time and its performance in bad weather while trying to detect traffic [9]. Artificial Intelligence (AI) has gained considerable prominence and has been increasingly applied across diverse domains [10].

One of the intriguing endeavors in vehicle tracking is the detection of anomalies in traffic offenses and negligent driving. The use of technology across several fields is becoming unavoidable, thanks to the development of new ITS technologies and the increasing desire for automation [11]. Automated vehicle traffic monitoring is one of the most significant applications being developed for speed or traffic management, offense detection, road tolls, and several other connected concerns [12]. A significant portion of the overall budget is used to handle these problems. It is quite challenging to conduct traffic monitoring in densely populated urban areas. There has been considerable success in lowering municipal occurrences via ITS mobility planning and traffic engineering applications [13]. Surveillance systems rely on traffic flow data, which consider important aspects like vehicle type, size, trajectory, and speed. These days, various traffic patterns may also be observed and recorded using vision-based devices.

While deep learning networks' training time is much slower on CPU computations, advancements in these networks have made ML-based models capable of accurately detecting a wide range of vehicles [14]. Improvements in technology, such as graphics processing units (GPUs) and tensor processing units (TPUs), have drastically reduced the training time. In several contexts, including smart self-governing, autonomous cars, intelligent observations, and smart city applications, DNNs have considerably outperformed traditional ML-based methods [15]. A sophisticated subfield of machine learning, DNNs built on neural networks come in handy when dealing with problems in various complicated models that are not always amenable to explanation using conventional statistical methods [16].

In addition, picture identification and classification heavily use CNNs, a deep neural network. These algorithms may recognize license plates, automobiles, persons, and many more things [17]. The ability of the

CNN to automatically extract important characteristics after training is one of its main advantages. Among the several CNN methods available, the most popular and widely used ones are R-CNN, Fast-RCNN, and Faster-RCNN [18]. However, devices with limited processing power and space still cannot process photographs due to the significant computational burden. For a long time, D-based algorithms were considered powerful picture identification techniques. There are many vehicle identification algorithms, but two of the most popular ones are CNN-based techniques, which fall into two categories: region-based and regression-based.

Vehicle detection and vehicle classification are related but distinct computer vision tasks. Vehicle detection involves identifying and localizing vehicles within an image or video frame, generally by predicting their spatial locations using bounding boxes together with their corresponding class labels. Vehicle classification, in contrast, assigns an input image or an already identified vehicle to a predefined category. In this study, the focus is specifically on vehicle image classification rather than object detection or localization. The evaluated deep learning architectures classify vehicle images into four categories: bicycle, bike, car, and truck. Since the proposed experiments do not estimate bounding boxes or vehicle locations, detection-specific measures such as Intersection over Union (IoU) and mean Average Precision (mAP) do not apply to the current experimental setup. Therefore, throughout the revised manuscript, we use the term "vehicle classification" when referring to the evaluated task, while "vehicle detection" is retained only when discussing the broader traffic-surveillance and urban-computing application context.

The aim of the present research was to design an accurate deep learning-based vehicle classification system that operates in an urban computing environment. For this purpose, the seven popular deep learning architectures were compared to each other in the task of vehicle classification into four classes: bicycle, bike, car, and truck, namely: VGG-16, VGG-19, Inception-V3, Inception-V4, ResNet-50, ResNet-101, and EfficientNet-B0. The training data was subjected to data augmentation methods to enhance the variety of the input samples and aid in model generalization. The experimental results show that EfficientNet-B0 outperforms ResNet-101 with an accuracy of 97.06%, which is the best classification accuracy. The comparative analysis has highlighted the relative strengths and performance differences of the CNN architectures evaluated and given practical insights into their appropriateness for vehicle classification in urban computing applications based on images. The results can be used to develop intelligent transportation and urban computing systems that rely on automated vehicle classification.

2. Related Works

Vehicle detection technology shares its fundamental objectives with target detection since it categorizes targets alongside determining their positions. Vehicle recognition algorithms exist in three principal categories: methods based on historical data, methods based on shallow machine learning, and methods based on deep learning models.

2.1. Traditional Vehicle Detection Algorithm

Manually designing target characteristics is required by traditional-based vehicle identification algorithms, just as it is by conventional target detection technologies [19]. The vehicle detection process uses preexisting information about vehicles by analyzing their lines, shadows, and edge features from their bodies. By analyzing shadow intensity between the bottom pixels and adjacent pixels, the system identifies the location of vehicles in the image and executes successful detection [20]. Nevertheless, lighting conditions will significantly affect this method's detection performance. This is because the gray value of the image is susceptible to light, and non-vehicle objects with a similar shape and size will cast shadows that are indistinguishable from the vehicle's, leading the algorithm to make an inaccurate judgment and rendering it unfit for high-precision detection [12]. Another way that characteristics of vehicles may be intentionally designed to identify them is by sensing their brake lights. While it is possible to accomplish vehicle recognition by sensing brake lights, the detection result is subpar owing to the method's severe constraints caused by the impact of light [14]. The previous knowledge-based detection technique relies on the vehicle's inherent qualities to complete the detection work. However, this approach has drawbacks, such as a low detection recognition rate, making it challenging to fulfill the mission's criteria [21].

Shallow machine-learning approaches remain an established method to recognize vehicles by joining diverse machine-learning algorithms that use vehicle characteristics to detect vehicles [22]. Shallow machine learning may increase detection accuracy by incorporating SVM and HOG features; nevertheless, this approach has certain limitations. A simple cascade enhances the detection accuracy while simultaneously scaling up the detection model and the computational cost. Even though using a shallow machine learning approach improves vehicle recognition accuracy somewhat, it is still feature selection dependent [23]. Its advancement has been limited mainly because of the vast modeling work required under complex and dynamic road conditions. The frame difference approach, the streamer method, and the backdrop modeling method are all examples of more conventional methods for vehicle identification [24]. Some identification effects may be achieved under specific settings using this algorithm, but it has several limits and is highly affected by ambient variables like lighting and weather [25]. As a result, it struggles to adapt to the ever-changing road traffic situation and fails to fulfill the real-time needs of vehicle detection.

2.2. A Deep Learning-Based Vehicle Detection System

The deep learning-based algorithms exist in two fundamental categories: identifying vehicles or targets. The detection technique with a single stage operates to determine whether an area qualifies as a candidate or recommendation [26]. The single network receives input images and generates the target bounding box and class label to determine the target identity within one stage. The networks integrate their operations to perform two-stage target identification, starting with pictures that create the proposed target regions. The classifier uses the generated regions for classification purposes.

R-CNN joins Fast R-CNN and Faster R-CNN among the established two-stage target detection algorithms [27]. The fundamental characteristic of this technique involves using a convolutional neural network to judge groups of candidate targets initially created from region proposals. The proposed areas undergo sparse sampling followed by feature extraction through a CNN, which leads to classification by SVM in R-CNN. The limitations of R-CNN have decreased, but it has delivered enhanced performance in target detection compared to conventional methods [28]. The detection rate problem is the biggest challenge among multiple problems affecting this system. Adding a spatial pyramid pooling operation between the convolution and fully connected layers in SPP-net reduces calculation information redundancy and network computational cost. The network requires this feature to extract independent features from multiple candidate regions, thus leading to several repetitions of computational operations. The system makes static image sizes by exclusively extracting features from incoming pictures [29]. The network performs its operations standardly, where candidate regions emerge from region suggestions. Then the CNN extracts image Roi features and ends with category assessment and position refinement to achieve object detection. The network gains better operational efficiency because the Roi feature information is extracted directly from feature maps, which makes the need for additional convolutions unnecessary [30].

Fast R-CNN achieves frame regression and classification training using the same network because it extends the SPP network optimization. The utilization of multi-task loss for classification replaces the support vector machine (SVM) in the system. The CPU executes conventional candidate area generation through standard techniques, which decreases the overall system processing speed. Target detection through end-to-end operation became truly possible after the release of Faster R-CNN. An RPN network removes candidate areas from the input, which can be shared with both the categorized CNN and the candidate regions generated during convolution. The first operation of Faster R-CNN produces multiple target candidate areas from the source. The system retrieves region characteristics through region mapping and spatial pyramid pooling. A CNN splits into multiple parallel branches, identifying various target features at the last stage. The architectural properties of RPN enable Faster R-CNN to generate regions faster than its preceding model, Fast R-CNN [31]. Before utilizing the multi-task loss function, the detector checks if candidate boxes have the proper features needed for detection, leading to the detection classification being assigned accordingly.

The detection methods used in the two stages function independently of procedures employed in single-stage detection. The one-stage target detection algorithm performs better than two-stage detection techniques since it simultaneously executes classification and regression calculations across various image areas.

According to references, single-stage target identification algorithms include the YOLO, YOLOv2, and YOLOv3 networks [32]. These detection systems transform the target detection work into a regression-based problem through end-to-end detection processing. A two-stage detection network within the YOLO network divides the input picture into grids so each grid detects targets that have their center points on it. The candidate frame in every grid undergoes replacement by anchor frames using the network structure. Real-time detection has become more effective, and the ability to detect objects has risen significantly because of this method. Expansion of the receptive field occurs due to enhanced connections between original convolution layers within YOLOv3 end-to-end detection methods, which results in superior small object detection capabilities for traditional YOLO networks [33]. The SE-TinyYOLOv3 algorithm represents an improved version of YOLOv3, which uses depth separation in its design structure. SE-TinyYOLOv3 decreases the model parameter count through architectural modifications that replace traditional convolution layers with depth-separable convolution layers in the YOLOv3 algorithm's feature extraction network. The network obtains deeper information about trained data features through its expanded lateral operational depth. The network system maintains assured detection accuracy and real-time detection by minimizing feature information loss while reducing parameter numbers.

Real-time vehicle detection systems need high-performance capabilities in operational road settings because of their complex and changing surroundings. The current target identification algorithms utilize regional recommendations for their operation while achieving high detection accuracy suitable for vehicle detection systems [34]. Its detection rate problems, combined with high computational requirements, limit this approach from satisfying heavy traffic condition requirements of highways. Detection speed drives the unmanned driving system to place more importance on it than other factors. The regression-based target identification method demonstrates superior detection speed and a well-developed structural design. The detection accuracy of this second approach falls below the initial technique yet provides superior real-time capabilities to the vehicle detection system. In practice-based applications, we usually employ a one-stage detection system for detecting vehicles. DarkNet-53 forms the basis for developing YOLOv3-based road vehicle target identification systems [35]. This original network structure enables engineers to develop neural networks with 30 convolutional layers. By implementing this network, the detection capabilities and training expenses have declined.

The original YOLOv3 serves as a base for an upgraded vehicle detection model to perform real-time road vehicle recognition among established areas of vehicle identification research. The increased network minimizes parameters, computation requirements, and model complexity using inverse residual networks for base feature extraction. The problem of gradient explosion and disappearance is resolved through this method. The improved network employs both group normalization and the softening no-maximum suppression technique as part of its strategy to decrease model inaccuracy caused by different batch data sizes. Such methods ensure immediate detection performance by addressing detection accuracy requirements. Research on the real-time capability of vehicle identification constitutes a key focus among investigations of intelligent transportation systems. Researchers have developed deep learning-based tracking systems to establish precise real-time solutions that adjust to various environments. Vehicle detection during real-time operation presents the main challenge for this design method [36-40].

3. Methodology and Data Collection

3.1. Proposed Architecture

Different CNN architectures perform training operations on the training and testing sets for VGG-16, VGG-19, Inception-V3, Inception-V4, ResNet-50, ResNet-101, and EfficientNet B0. The educational phase of these models took place on the 1.3 million-item ImageNet ILSVRC dataset that contained 1000 picture sets for training, 100,000 images for research, and 50,000 images for validation. Implementing batch normalization layer settings results in superior outcomes for reducing validation loss and improving validation and training accuracy. Batch Normalization is a technique to normalize deep neural network layer inputs throughout any selected mini-batch. The training stability created by this approach shortens the deep network development

process by decreasing the required training iterations. All deep learning models use the Categorical Cross-Entropy (CCE) loss function. The optimization method deploys Adam's algorithm. Adam performs differently from stochastic descent in a traditional manner. The learning rate, referred to as alpha, remains static throughout training steps in stochastic gradient descent. Adam incorporates benefits from the Adaptive Gradient Algorithm (AGA) and RMSProp by serving as two additional variants of stochastic gradient descent.

CNNs are trained on Colab, a Google service for training and evaluating machine learning, deep learning, and other AI-related implementations in the cloud. A Tesla T4 GPU, provided by Google Colab, is used for our classification task. After training, the validation process is carried out by predicting the class of the vehicles (car, bike, truck, bicycle), as shown in Fig. Implementation details and results are discussed below. For each CNN classifier, F1-Score, Recall, and Precision are calculated using the following formulas.

$$\text{F1Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP}} + \text{FN} \quad (2)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}} + \text{FP} \quad (3)$$

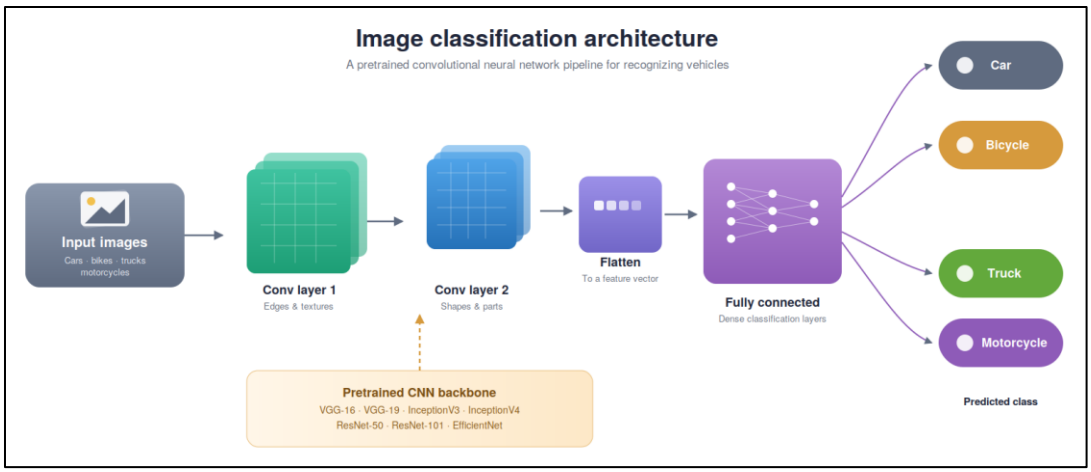


Figure 2. Vehicle Detection in an Urban Computing System Using a Convolutional Neural Network Block Diagram

3.2. Dataset Collection and Data Partitioning

The dataset consists of four classes of RGB vehicle images: bicycle, bike, car, and truck. Images were collected from Google Images, while truck images were additionally obtained from Kaggle. The images were organized into separate training and test directories before model training. All images were resized to the required input dimensions of 224 × 224 pixels for the CNN models.

To increase the diversity of the training samples, data augmentation was applied dynamically only to the training data using an ImageDataGenerator. The augmentation procedure included rescaling pixel values by 1/255, a shear range of 0.2, a zoom range of 0.2, and horizontal flipping. In contrast, the test-data generator applied only pixel-value rescaling and did not perform geometric augmentation. Therefore, augmented training samples were not directly added to the test directory.

The training generator used a batch size of 32 and categorical class labels. During model training, the designated test generator was supplied as the validation data to monitor model performance. After training, the test data were used to generate the final predictions, confusion matrices, precision, recall, and F1-score values. This distinction is explicitly described in the revised manuscript to avoid ambiguity between validation monitoring and final testing. The sample of the photographs is shown in Fig.

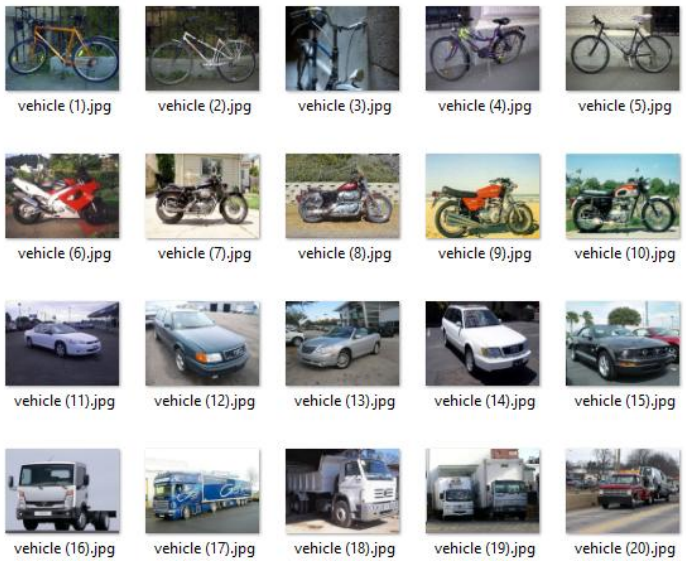


Figure 3. Sample Dataset of The Different Types of Vehicle Detection

3.3. Training, Validation, and Data Augmentation

The CNN models were trained using a designated training dataset, with data augmentation applied dynamically during training. The augmentation procedure included pixel-value rescaling by \(1/255\), a shear range of 0.2, a zoom range of 0.2, and horizontal flipping. In contrast, the designated test-data generator applied only pixel-value rescaling and no geometric augmentation. Therefore, augmented images were generated dynamically during training and were not explicitly written into the test directory.

In the implemented experimental procedure, the designated test generator was also supplied as the validation data during model training. Accordingly, the epoch-wise validation performance corresponds to the performance of this designated test set, rather than to an independent validation subset. After completion of training, the same designated test set was used for the final evaluation, including accuracy, precision, recall, F1-score, and confusion-matrix analysis. This procedure is now explicitly stated to avoid incorrectly implying that an independent validation set was used.

The epoch-wise training results provide information about the optimization behavior of the models. For example, EfficientNet-B0 showed a progressive improvement in reported accuracy from 0.9471 to 0.9706 and a reduction in loss from 0.2895 to 0.0102 over five epochs. Similar epoch-wise loss and accuracy results are reported for the VGG, Inception, and ResNet architectures.

Because the test set was used as validation data during training, the present experiment does not provide a fully independent validation-based assessment of overfitting. Therefore, the observed training curves should be interpreted as evidence of learning behavior rather than as definitive proof of the absence of overfitting.

3.4. Dataset Collection and Source Considerations

The data is made up of RGB images of 4 classes of vehicles: bicycle, bike, car, and truck. The images of vehicles were taken from Google Images by searching for the categories, and other truck images came from Kaggle. Images obtained were manually checked, sorted by the respective vehicle class, and labeled prior to being included in the experimental data set. The images were then resized to the input dimensions needed for the CNN architectures and preprocessed and augmented using the methods detailed below.

Google Images is not a repository of curated images, but rather an image search service; thus, image provenance and licensing concerns arise when using images found through Google Images. The addition of an image to Google Images doesn't guarantee that the image is free for redistribution or unrestricted reuse. As such, the current study does not presume that every image retrieved has the same licensing requirements. Images were used for research and experimentation, and it is recognised that a limitation of this dataset is that it is not complete in terms of image-level provenance and licensing metadata. Any future construction of data

sets will focus on data sets where reuse permissions are clearly documented and will include source and licensing metadata where available.

Data should also be seen in the context of lack of diversity and representativeness of sources. The images found on the web can differ significantly in many aspects such as geographical location, vehicle models, camera angles, lighting, background complexity, image quality, and environmental conditions. Therefore, the dataset might not always reflect the diversity of vehicles and visual conditions that are subjected to real-world urban traffic surveillance. Thus, the results represent the classification performance of the tested CNN configurations using the dataset created, and not the performance of the tested CNN in all urban environments. A larger, more geographically diverse, and independently captured real-world traffic image set will be used in the future to enhance the generalizability and representativeness.

4. Results and Discussion

The global population is constantly moving towards urbanization. Almost two-thirds of the global population will dwell in urban areas within the next three decades. Therefore, community planners need solutions for successful land use and traffic control. Acquiring accurate, timely, and detailed data is essential for creating significant urban infrastructure and intelligent cities. The field of urban computation and intelligence includes creating techniques for pervasive sensing, cognitive computing, interactive connectivity, and mass data processing technologies that provide innovative ways to transform urban landscapes, living standards, and intelligent urban systems. Table 1 shows Parameters for the Inception-V3, Inception-V4, VGG-16, VGG-19, ResNet50, ResNet101, and Efficient-Net B0 models are described in depth.

Table 1. Parameters Detail of Inception-V3, Inception-V4, VGG-16, VGG-19, ResNet50, ResNet101, Efficient-Net B0

Parameter	Value
Data organization	Training set and test set
Training augmentation	Rescaling, shear, zoom, horizontal flip
Test preprocessing	Rescaling only
Input image size	224 × 224
Batch size	32
Number of classes	4
Classes	Bike, Bicycle, Truck, Car
Validation during training	Designated test set

4.1. The Evaluated Results of the VGG-16 and VGG-19 Models

Table 2 shows the detailed performance of the two models (VGG-16 and VGG-19) during training, specifically recording the model's loss and accuracy after every 10 epochs. The epochs correspond to the number of times the models have been trained on the full training dataset, which gives us the opportunity to check the progress of the models' learning. The table then presents both models' loss and accuracy values, which are important for the model's predictive power. Loss is the error in the model's prediction; if the prediction is closer to the actual value, the lower it is, the better the model is. On the contrary, the accuracy values show the ratio of correctly classified images, with a 'higher' value of accuracy which means 'higher' accuracy of the model. From the table, however, the loss and accuracy changes suggest the dynamics of training for each model.

Table 2. Loss and Accuracy Performance of VGG-16 and VGG-19 Models by Training and Testing Dataset

VGG-16			VGG-19		
Epoch	Loss	Accuracy	Epoch	Loss	Accuracy
1	0.0685	0.9459	1	0.3804	0.9482
2	0.2891	0.9459	2	0.0701	0.9471
3	0.0932	0.9259	3	0.2347	0.9388
4	0.1851	0.9600	4	0.0627	0.9624

5	0.0775	0.9541	5	0.0625	0.9271
6	0.1154	0.9506	6	0.1571	0.9412
7	0.1967	0.9435	7	0.0091	0.9565
8	0.1337	0.9576	8	0.0390	0.9447
9	0.0280	0.9635	9	0.3807	0.9565
10	0.2145	0.9635	10	0.1028	0.9600

The final evaluation results of performance on the test dataset of the VGG-16 and VGG-19 models are summarized in Table 3. This is the tabular form of the final loss and accuracy derived by the models, in a compact form of the overall effectiveness of each model. However, it clearly demonstrates that the loss value drops to 0.2145 with 96.35% accuracy for VGG-16 and drops to 0.1028 with 96.00% accuracy for VGG-19. These values directly compare the performance of the models and will tell you a bit about how closely the models predict and how much they get wrong. Obviously, the VGG-19 model had a much lower loss, which indicates that it is perhaps more appropriate for the data, but the VGG-16 has slightly higher accuracy.

Table 3. The Evaluated Loss and Accuracy of VGG-16 and VGG-19 Models

Model Name	Loss	Accuracy
VGG-16	0.2145	96.35%
VGG-19	0.1028	96.00%

In Fig, the data from Table 2 and Table 3 are depicted graphically. Clearly compares the loss and accuracy of VGG-16 and VGG-19 with the aid of bar plots or other graphically similar elements. Changes in loss and accuracy are plotted over training epochs, and the final evaluation of the test set for all models' performance is shown in the Fig. Graphically presenting the models can aid rapid and easy model performance calculations and provide instant insight into the success of the models and the patterns of loss and accuracy as the models are trained.

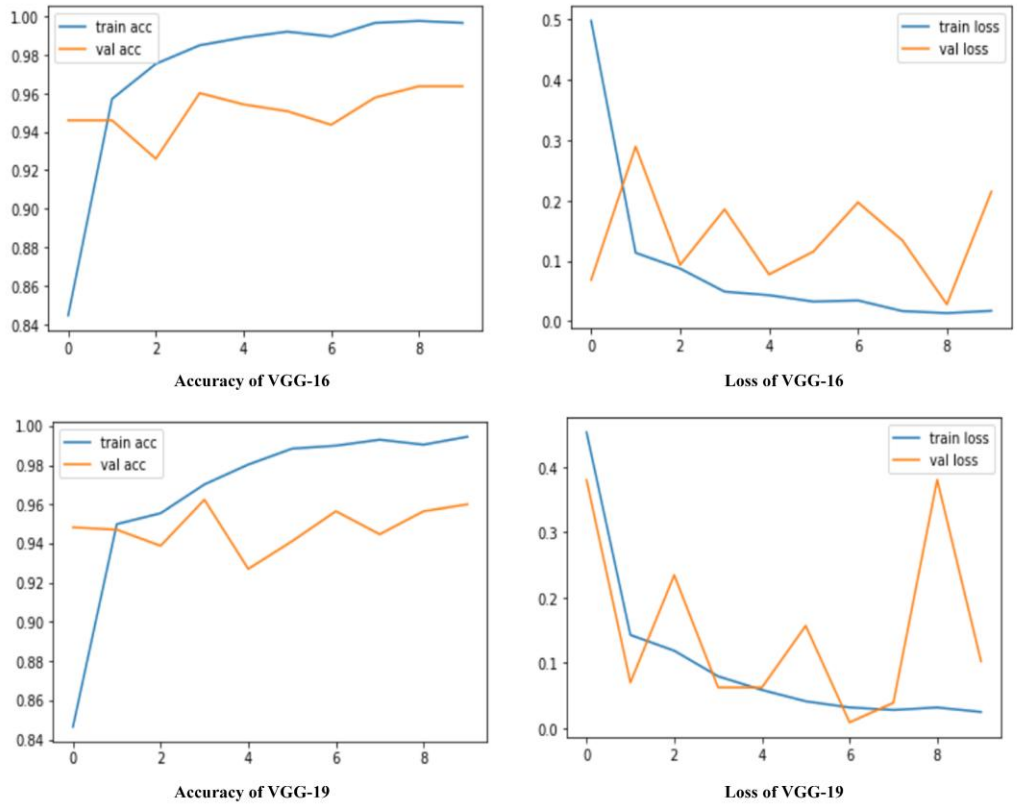


Figure 4. VGG-16 and VGG-19 Loss and Accuracy for Training and Testing of Vehicle Dataset

Instead of simply looking at the accuracy, Table 4 provides the precision, recall, and F1 score for each vehicle category, with Bicycle and Bike showing a high level of accuracy and Truck being more complex, showing

varying strengths and weaknesses across categories and models, giving a deeper understanding of the intricate capabilities and constraints of each model in classifying specific types of vehicles.

Table 4. The F1-Score, Recall, and Precision Are Calculated for Each Predicted Class Of VGG-16 and VGG-19

Categories	VGG-16			VGG-19		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bicycle	0.9000	0.9818	0.9391	0.9444	0.9272	0.9357
Bike	0.9897	0.9633	0.9763	0.9761	0.9533	0.9645
Car	0.9880	0.9705	0.9792	0.9910	0.9764	0.9837
Truck	0.8846	0.9200	0.9019	0.8421	0.9600	0.8971

4.2. The Evaluated Results of the Inception-V3 and Inception-V4 Models

The training performance over epochs of the Inception-V3 and Inception-V4 networks is presented in Table 5, where the loss and accuracy at each of the five training epochs are highlighted. It shows how the dynamic learning process of both models works through variations in the loss and accuracy functions across the epochs, showing how the models adapt to the training data. We note that Inception-V3 has quite a lot of variability in loss (especially a very large spike in epoch three and a massive drop in epoch 4) while retaining generally high accuracy. On the other hand, the loss trajectory of Inception V4 is more consistent, although this comes at the cost of accuracy on the last epoch. The insight we can gain from this table regarding the training dynamics of both models gives us a way to determine the strengths and weaknesses of both models throughout the learning process.

Table 5. Loss and Accuracy Performance of Inception-V3 and Inception-V4 by Training and Testing Dataset

Inception-V3			Inception-V4		
Epoch	Loss	Accuracy	Epoch	Loss	Accuracy
1	1.8973	0.9176	1	1.2344	0.9471
2	0.0699	0.9282	2	0.2030	0.9565
3	4.9217	0.8659	3	0.1394	0.9482
4	0.0035	0.9459	4	0.1660	0.9506
5	0.1908	0.9518	5	0.4999	0.9353

The final evaluation results obtained on the designated test set for the two models, Inception-V3 and Inception-V4, are synthesized in Table 6. Finally, it summarizes the models' effectiveness by providing the final accuracy and loss values. Inception-V3 achieved a loss of 0.1908 and an accuracy of 95.18%, whereas Inception-V4 achieved a loss of 0.4999 and an accuracy of 93.53%. These figures directly compare the different performance of the models on the test data, showing that Inception-V3 performs better than Inception-V4.

Table 6. The Evaluated Loss and Accuracy of Inception-V3 and Inception-V4 Models

Model name	Loss	Accuracy
Inception-V3	0.1908	95.18%
Inception-V4	0.4999	93.53%

Finally, the data in Table 5 and Table 6 are visually represented in Figure 1 to enable a comparative study between the Inception-V3 and Inception-V4 models. It shows the training epochs, where changes in loss and accuracy are shown through graphical elements, as well as the final evaluation results on the test dataset. Visualizing this enables rapidly obtaining a sense of how the models are performing, quickly and intuitively comparing effectiveness, ss, and observing the trends in loss and accuracy over the time of training.

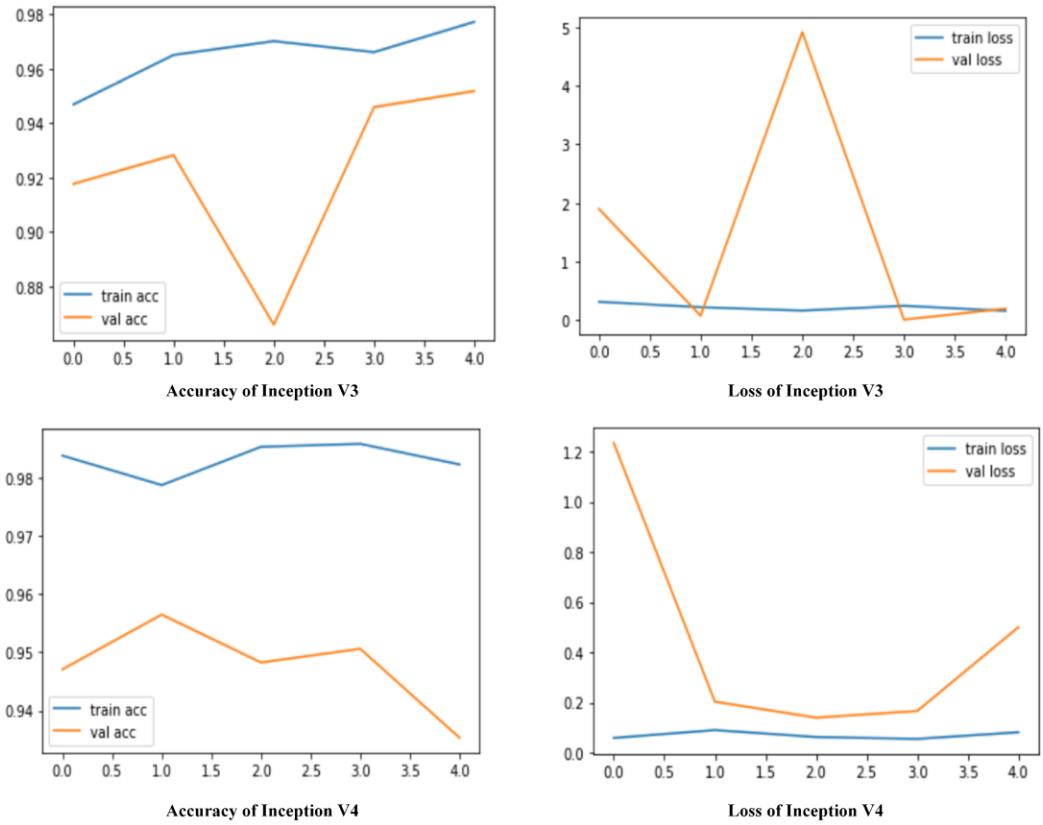


Figure 1. Inception-V3 and Inception-V4 Training and Testing Accuracy and Loss

The results of the classification performance of both Inception-V3 and Inception-V4 are also detailed in Table 7, showing the precision, recall, and F1-score for each vehicle category; there are subtle differences in the ability of both networks to categorize. Inception-V3 has been performing consistently and well in all categories, especially in car classification. Meanwhile, Inception-V4 achieves superb performance in bicycle and car recognition. It does show, however, a marked difference in accuracy when it comes to truck classification, meaning that it is prone to misclassifying other types of vehicles, revealing strengths and weaknesses in each model when it comes to correctly identifying a vehicle's type.

Table 7. The F1-Score, Recall, and Precision Are Calculated for Each Predicted Class of Inception-V3 and Inception-V4

Categories	Inception-V3			Inception-V4		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bicycle	0.9693	0.8636	0.9134	0.9059	0.9636	0.9339
Bike	0.9305	0.9833	0.9562	0.9929	0.9366	0.9637
Car	0.9821	0.9705	0.9763	1.00	0.9088	0.9522
Truck	0.8989	0.8900	0.8946	0.7021	0.9900	0.8215

4.3. The Evaluated Results of ResNet-50 and ResNet-101 Models

Training performance over five epochs was presented for ResNet-50 and ResNet-101 in terms of loss and accuracy in Table 8. The loss of ResNet-50 fluctuates wildly, spiking in epochs 2, 3, and 4, with a sharp decrease in epoch 5, and its accuracy is generally high but fluctuating. On the other hand, ResNet's loss value is consistent throughout the training, and accuracy remains consistent with a slight variation, achieving a maximum value in epoch 5. The dynamic views of the change in the models' performance during the training are quite evident from this table.

Table 8. Loss and Accuracy Performance of ResNet-50 and ResNet-101 Models by Training and Testing Dataset

ResNet-50			ResNet-101		
Epoch	**Loss**	**Accuracy**	**Epoch**	**Loss**	**Accuracy**
1	1.6367	0.8576	1	1.4914	0.9282
2	6.9615	0.9024	2	0.4740	0.9141
3	3.4575	0.9047	3	0.6668	0.9576
4	3.0807	0.9259	4	0.6991	0.9294
5	0.0684	0.9529	5	0.1811	0.9682

The final evaluation results of ResNet-50 and ResNet-101, concerning the performance on the test dataset, are summarized in Table 9. The loss is 0.0684, and accuracy is 95.29% for ResNet-50, and 0.1811 and 96.82% for ResNet-101. For this overall effectiveness, this table also gives a concise overview comparing the models to see which one will perform best. From here, we can see that the ResNet-101 model will have better accuracy but a higher loss when tested.

Table 9. The Evaluated Loss and Accuracy of the ResNet-50 and ResNet-101 Models

Model name	Loss	Accuracy
ResNet-50	0.0684	95.29%
ResNet-101	0.1811	96.82%

Figure 2 shows Table 8 and Table 9 as a 2x2 matrix for directly comparing ResNet-50 and ResNet-101's performance. It probably shows loss vs. training epochs and accuracy vs. training epochs, particularly the final value on the test dataset, using multiple graphical elements. This visualization allows for quick and intuitive understanding of the model's performance in order to be able to easily compare the effectiveness of the model and see the trends in the loss and accuracy of the model during the training process.

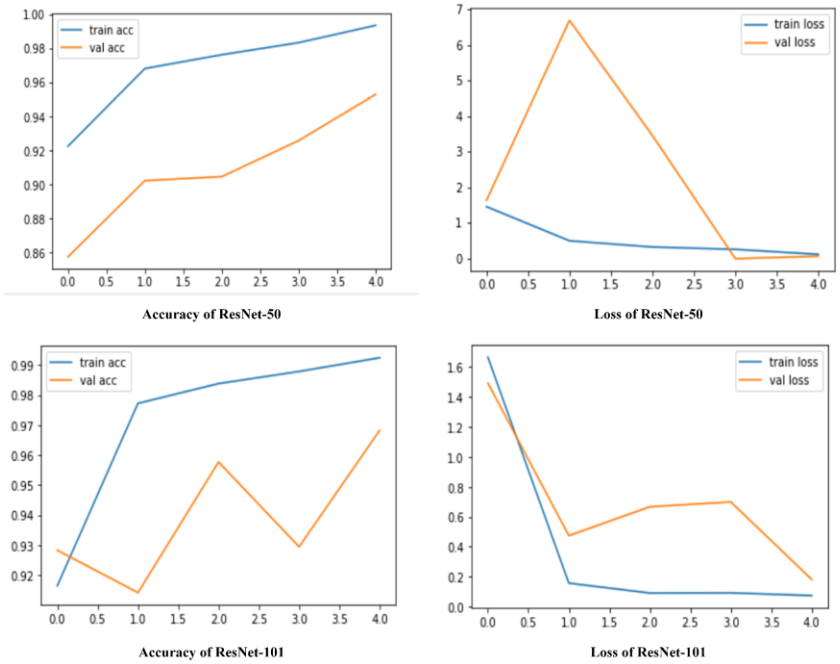


Figure 2. ResNet-50 and ResNet-101 Loss and Accuracy for Training and Testing of Vehicle Dataset

Table 10 presents detailed comparison results of the classification capabilities of ResNet-50 and ResNet-101 on each of the vehicle categories, including precision, recall, and F1-scores, revealing their respective strengths and weaknesses. Overall, ResNet-50 has very good performance and is superior compared to other categories, particularly in the car and bike classification. ResNet-101 also achieves top performance in the car and truck identification tasks. It does have a very noticeable lack of bicycle and bicycle classification, however, with poor

recall and high precision, respectively, showing the different capabilities of which each model is able to accurately distinguish between the various vehicle types.

Table 10. The F1-Score, Recall, and Precision Are Calculated for Each Predicted Class of ResNet-50 and ResNet-101

Categories	ResNet-50			ResNet-101		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bicycle	0.8515	0.9909	0.9159	0.8870	0.1000	0.9401
Bike	0.9895	0.9466	0.9676	0.1000	0.9433	0.9708
Car	0.9908	0.9529	0.9715	0.9969	0.9764	0.9866
Truck	0.8611	0.9300	0.8942	0.8909	0.9800	0.9333

4.4. The Evaluated Results of the EfficientNet B0 Model

The training evolution of the EfficientNet B0 model for five epochs is reported in Table 11, where the loss and the accuracy values are calculated at each epoch. The loss curve clearly shows a decline in loss from epoch 1 to epoch 5, indicating that the model is getting better as it continues to be trained. Meanwhile, the accuracy also greatly improves, which means the model has learned correctly and has higher accuracy. The model has been successfully trained, as the loss value and the accuracy value of the fifth epoch are low. This table explicitly outlines the model's performance on the training data to show it learned from and adapted to this dataset.

Table 11. Loss and Accuracy Performance of EfficientNetB0 Model by Training and Testing Dataset

Epoch	Loss	Accuracy
1	0.2895	0.9471
2	0.2633	0.9565
3	0.1207	0.9624
4	0.0679	0.9647
5	0.0102	0.9706

This near-perfect accuracy in the test set, with just a 0.0512 increase in the loss function from the training results (see Table 12), may mean that the model does not overfit too much to the training data and thus has a strong transfer capability and is highly applicable in real-world environments.

Table 12. The Evaluated Loss and Accuracy of EfficientNet B0 Model

Model Name	Loss	Accuracy
EfficientNetB0	0.0102	97.06%

Figure 3 is a graphical representation of the training data in Table 11 and possibly the test data in Table 12. The graph or chart shows the loss and accuracy as the model is trained over five epochs, highlighting the model's learning curve. If there is test data, it will display the final loss and accuracy on the test data. This visual aid enables a quick and intuitive understanding of the EfficientNet B0 model's training and, potentially, testing performance, allowing for easy observation of the trends in loss and accuracy.

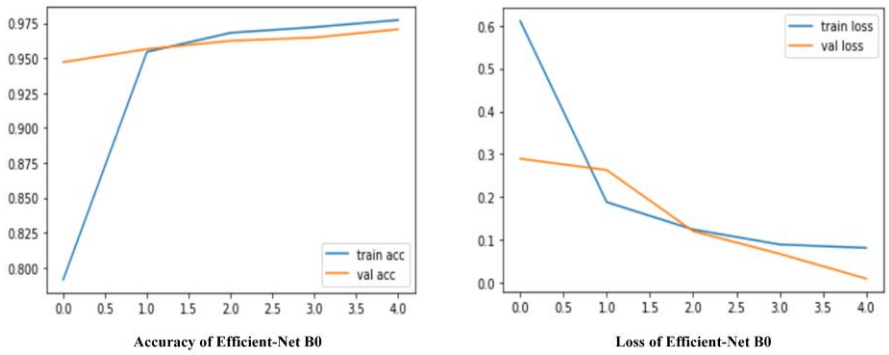


Figure 3. EfficientNet B0 Loss and Accuracy for Training and Testing of Vehicle Dataset

Precision, recall, and F1-score are used to calculate a macro and weighted average in the EfficientNet B0 model, as shown in Table 13.

Table 13. The F1-Score, Recall, and Precision Are Calculated for Each Predicted Class of EfficientNetB0

Categories	Precision	Recall	F1-Score
Bicycle	0.9159	0.9909	0.9519
Bike	0.9965	0.9566	0.9761
Car	0.9881	0.9794	0.9837
Truck	0.9056	0.9600	0.9320

4.5. Comparison Results of All Deep Learning Models

The results of seven deep learning models are summarized in Table 14, where it can be seen that EfficientNet B0 has the lowest loss of 1.02% and the highest accuracy of 97.06%, which proves the high effectiveness of this model in vehicle type classification. Meanwhile, Inception-V4 has the worst accuracy, the highest loss, and is in the least developed state. Other models like ResNet-50, ResNet-101, and the VGG models are able to achieve high accuracy and low loss, indicating the different capacity of each model to correctly classify the type of vehicles.

Various measures such as accuracy, precision, recall, F1-score, and confusion matrix were used to assess the performance of the vehicle-classification models in order to get a comprehensive assessment of the models. Accuracy: Percentage of images correctly classified out of the total number of images evaluated. Precision represents the rate of correctly classified samples to the rate of samples predicted as a given vehicle category, while recall represents the rate of correctly identified samples to the rate of actual samples in a given vehicle category. The F1-score is the harmonic mean of the precision and recall.

Aside from the overall accuracy, precision, recall, and F1-score were also computed for each class for the four classes of vehicles (bicycle, bike, car, truck). This class-wise analysis is useful for a more detailed determination of consistent performance of a model from one vehicle class to the other. The confusion matrices also help to show which predictions are correct, which are incorrect, and to see how the different classes are misclassified.

The results show that the models have high classification performance for the vehicle categories tested. The overall accuracy of EfficientNet-B0 was the highest at 97.06%, EfficientNet-50 at 96.82%, and VGG-16 at 96.35%. For EfficientNet-B0, the class-wise results show precision/recall/F1-score values of 0.9159/0.9909/0.9519 for bicycle, 0.9965/0.9566/0.9761 for bike, 0.9881/0.9794/0.9837 for car, and 0.9056/0.9600/0.9320 for truck. The results suggest that the model is performing well in all four classes and that the bicycle and truck classes have relatively low precision.

The confusion matrices are also shown for all seven evaluated architectures in addition to the numbers. Accuracy, together with precision, recall, F1-score, and confusion-matrix analysis, can give a more complete characterization of the classification performance than accuracy alone.

Table 14. The Statistical Comparison Inception-V3, Inception-V4, VGG-16, VGG-19, ResNet50, ResNet101, Efficient-Net B0 Models Based on Loss and Accuracy

Model name	Loss	Accuracy
VGG-16	21.45%	96.35%
VGG-19	10.28%	96.00%
Inception-V3	19.08%	95.18%
Inception-V4	49.99%	93.53%
ResNet-50	6.84%	95.29%
ResNet-101	18.11%	96.82%
EfficientNet-B0	1.02%	97.06%

4.6. Computational Efficiency Analysis

In addition to classification performance, computational efficiency was also evaluated to enable a comprehensive comparison of the CNN architectures that were evaluated. The number of trainable parameters, model size, and inference time per image were taken into account. The number of trainable

parameters is a measure of how complex each network is, while the model size is a measure of the storage and memory needed for each model. To make a fair comparison, the same input image dimensions and computational environment were used for all the models to measure inference time. The FPS was computed as (FPS = 1000/t), with (t) being the average time in milliseconds per image required for an inference.

The computational results illustrate the cost and the accuracy for the different architectures being evaluated. Competitive classification performance can be achieved in larger architectures, but in general, larger architectures are more intensive in computation and memory. The EfficientNet-B0 model recorded the best accuracy of 97.06% in classification accuracy, which is higher than ResNet-101 with 96.82% and VGG-16 with 96.35%. The computational-efficiency measurements give further proof for the assessment of the applicability of these architectures to real-world urban-computing applications.

Thus, model selection should take into account the computational demands as well as prediction capability. This joint analysis allows a more precise description of a model as efficient, while not just considering the accuracy of the classification.

4.7. Confusion Matrices of All the Trained CNN Architectures

Confusion matrices are a visualization approach used here; a confusion matrix displays the number of correct classifications (diagonal cells) and the number of wrong classifications (off-diagonal cells), making it easy to directly compare the strengths and weaknesses of each model's ability to classify each vehicle type by comparing the degree of accuracy for each model; this visualization approach is shown in Figure 4.

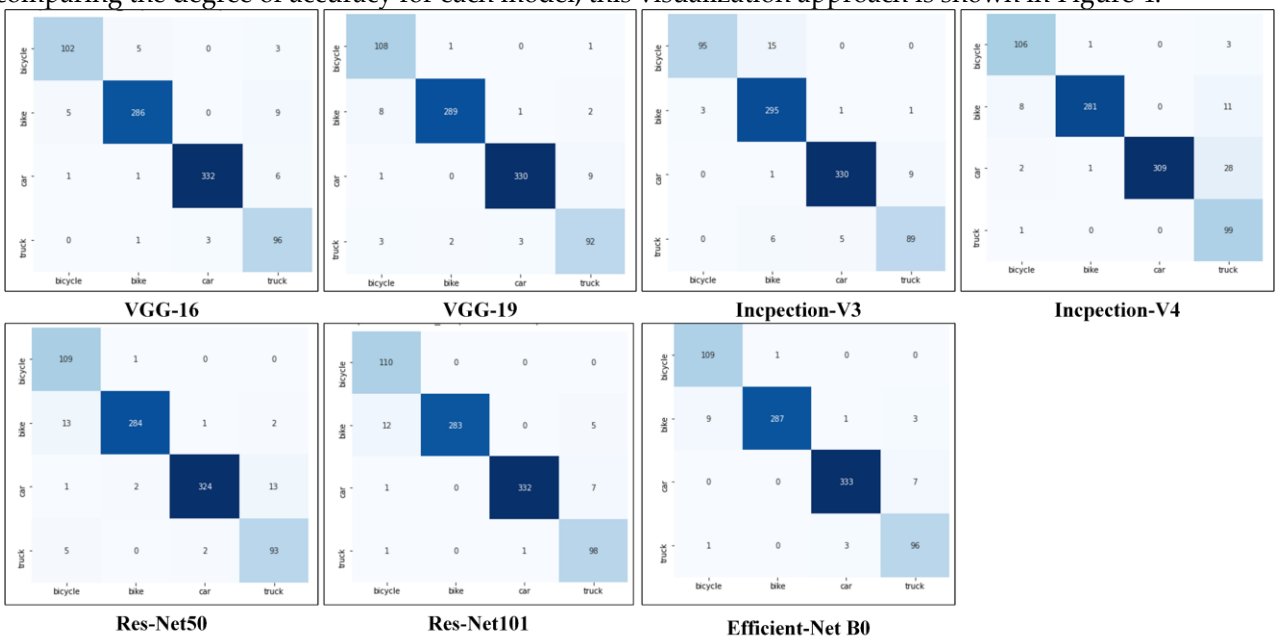


Figure 4. Confusion Matrices Showing Classification Performance of VGG-16, VGG-19, Inception-V3, Inception-V4, ResNet-50, ResNet-101, and Efficient-Net B0 Models

4.8. Robustness Under Real-World Urban Conditions

While all the CNN models evaluated are able to attain high classification accuracy on the dataset of vehicle images that was created, real-world urban environments could present significantly more visual variability when deployed. Vehicle occlusion, illumination variation, and camera-viewpoint variation can have an impact on the visibility or appearance of discriminative vehicle parts.

In urban traffic scenarios, occlusion is a prevalent issue that arises as vehicles may be partially obscured by other vehicles, pedestrians, roadside elements, or environmental objects. Partial visibility can diminish the information that may be used by the classifier for the class and can increase the level of confusion between visually similar classes. This dataset does not include a controlled occlusion benchmark, and so it is not possible

to make a quantitative determination regarding the robustness of this dataset under varying levels of occlusion from the experiments presented here.

Variation of illumination is another significant challenge. Bright daylight, shadow, low-light, nighttime, and glare environments may show significant variations in contrast and colour distribution of images. The augmentation strategy employed in the present study involves rescaling, shear, zoom, and horizontal flipping but fails to explicitly simulate changes of brightness, contrast, and night illumination. Thus, performance should be considered within the visual context that is provided by the experimental data set.

Camera viewpoint can also be a factor affecting the performance of classification. The traffic monitoring cameras in the city can shoot vehicles from different angles, such as a front, rear, side, elevated, or oblique angle. These differences may alter the external form and size of vehicle parts and can thus have an impact on classification. The impact of viewpoint variation has not been measured independently from the current dataset, as it was not built to be controlled multi-view.

The above indicates that the reported accuracy may not be a true reflection of the robustness of the system under all urban traffic conditions. However, the accuracy of the models was satisfactory, with the best reported being 97.06% for the EfficientNet-B0 model and 96.82% for ResNet-101. The results are thus a useful baseline for vehicle classification, and the next step is to validate them in the realistic conditions of operational urban traffic surveillance systems.

5. Conclusion and Future Work

In this work, the seven commonly used deep-learning architectures, namely Inception-V3, Inception-V4, VGG-16, VGG-19, ResNet-50, ResNet-101, and EfficientNet-B0, were implemented and compared in an urban computing environment for vehicle-type classification. The classification loss, accuracy, precision, recall, and F1-score were used to evaluate the models. Of all the evaluated architectures, EfficientNet-B0 had the highest classification accuracy of 97.06%, ResNet-101 had 96.82%, and VGG-16 had 96.35%. The remaining models achieved accuracies of 96.00% for VGG-19, 95.29% for ResNet-50, 95.18% for Inception-V3, and 93.53% for Inception-V4. The results suggest that EfficientNet-B0 was the best-performing architecture for classification among the ones studied in this work.

But the accuracy of classification is not the only way to ensure computational efficiency. Thus, computational efficiency should also be taken into account along with predictive performance, especially for urban computing applications where computational resources, memory, and processing latency might play a role. The number of trainable parameters, the size of the model, and the time required for inference under a fixed computational environment can be used to assess the complexity of the model in the updated assessment. This gives them a wider foundation on which to make comparisons with other architectures, as well as assessing how well a model will work for actual deployment.

In addition, the results show that the classification performance for the examined data set does not necessarily improve with increasing/deeper architectures. For instance, ResNet-101 had 96.82% accuracy, and EfficientNet-B0 had 97.06% accuracy. Thus, the selection of models should take into account the accuracy and the computational time of the models, instead of only the depth of the model.

There are some areas of future research. First of all, it is important to make the dataset more extensive by adding more and a wider range of vehicle images, such as those captured in various weather, lighting, camera perspectives, road and traffic conditions. This growth may help to enhance the generalizability and robustness of the classification models. Secondly, future research could test the models with separately collected real-world traffic-camera data to assess their performance in real urban scenarios. Third, vision architectures can be explored and compared to the CNN architectures tested here, such as transformers and more recent vision architectures based on transfer learning.

Also, future research is needed to explore deployment on resource-limited edge devices. An important step is to evaluate the inference latency, memory requirements, model size, energy consumption, and throughput systematically to decide if the chosen architecture is appropriate for real-time urban traffic applications. Other

compression methods such as pruning, quantization, and knowledge distillation could be explored to reduce the computational resources required without compromising the classification accuracy.

Lastly, computer-vision-based traffic monitoring introduces potential bias into the datasets, class imbalance, and environmental variability, as well as ethical and privacy concerns, which should be addressed in future studies. These are especially significant if the models are used in an urban system. In conclusion, the results presented in this work illustrate the ability of deep-learning architectures to classify vehicles by type and the crucial need to balance predictive performance and computational resources when selecting models for urban computing tasks.

One drawback of the current work is that there was no validation set that was separated from the training set when evaluating the final results. The validation data used for model training were the same as the designated test data, and the results of the models were compared to the designated test data for the final performance evaluation. Data augmentation was performed on the training generator, but not on the test generator; the use of the test set for validation monitoring can create a level of evaluation bias and can reduce the rigor of the conclusions about the ability of the model to generalize and the overfitting problem.

The proposed models will be tested in future with a separate set of real-world traffic camera data that includes varying lighting, weather, camera viewpoint/angle, vehicle density, and occlusion. Additionally, specific augmentation methods such as illumination and partial-occlusion simulations will be explored to enhance the system's ability to cope with difficult urban scenarios. Furthermore, the potential for using domain adaptation and transfer learning to narrow the performance gap between curated sets of images and actual traffic situations could be considered. Such experiments will give a more stringent test of the ability to generalize the proposed vehicle-classification approach in a reliable fashion to practical urban-computing applications.

Funding Statement: This article has been funded by the Deanship of Scientific Research in King Faisal University with Grant Number KFU264161. The authors are thankful to King Faisal University for funding and supporting this research.

References

1. A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Vehicle detection from aerial images using deep learning: A comparative study," Electronics, vol. 10, no. 7, p. 820, 2021.
2. A. Farid, F. Hussain, K. Khan, M. Shahzad, U. Khan, and Z. Mahmood, "A fast and accurate real-time vehicle detection method using deep learning for unconstrained environments," Applied sciences, vol. 13, no. 5, p. 3059, 2023.
3. R. M. A. Latif, M. Umer, N. Jamal, Y. Zhao, F. Ullah, and J. Ahmad, "Real-Time Helmet Violation Detection in Traffic Surveillance Using Optimized Faster R-CNN with Environmental Robustness," in International Conference on Information Systems and Computing Technology, 2025, pp. 62-71: Springer.
4. Almalkawi, I. T., Halloush, R., Alsarhan, A., Al-Dubai, A., & Al-Karaki, J. N. (2019). A lightweight and efficient digital image encryption using hybrid chaotic systems for wireless network applications. Journal of Information Security and Applications, 49, 102384.
5. Neupane, B., Horanont, T., & Aryal, J. (2022). Real-time vehicle classification and tracking using a transfer learning-improved deep learning network. *Sensors*, *22*(10), 3813.
6. S. Kumar, E. Rajan, and S. Rani, "A study on vehicle detection through aerial images: Various challenges, issues and applications," in 2021 international conference on computing, communication, and intelligent systems (ICCCIS), 2021, pp. 504-509: IEEE.
7. C. Chen, C. Wang, B. Liu, C. He, L. Cong, and S. Wan, "Edge intelligence empowered vehicle detection and image segmentation for autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 11, pp. 13023-13034, 2023.
8. Z. Wang, J. Zhan, C. Duan, X. Guan, P. Lu, and K. Yang, "A review of vehicle detection techniques for intelligent vehicles," IEEE Transactions on Neural Networks Learning Systems, vol. 34, no. 8, pp. 3811-3831, 2022.
9. R. M. A. Latif, N. Jamal, U. Raza, F. Ullah, L. Mostarda, and D. Cacciagrano, "Blockchain-Secured Deep Learning Pipeline for Real-Time Medical Diagnostics on Edge Devices," in International Conference on Frontiers of Electronics, Information and Computation Technologies, 2025, pp. 356-373: Springer.
10. Alhijawi, B., Kilani, Y., & Alsarhan, A. (2020). Improving recommendation quality and performance of genetic-based recommender system. International Journal of Advanced Intelligence Paradigms, 15(1), 77-88.
11. M. R. Ibrahim, J. Haworth, and T. Cheng, "URBAN-i: From urban scenes to mapping slums, transport modes, and pedestrians in cities using deep learning and computer vision," Environment Planning B: Urban Analytics City Science, vol. 48, no. 1, pp. 76-93, 2021.
12. U. Mittal, P. Chawla, and R. Tiwari, "EnsembleNet: A hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models," Neural Computing Applications, vol. 35, no. 6, pp. 4755-4774, 2023.
13. Y. Zhang, Y. Sun, Z. Wang, and Y. Jiang, "YOLOv7-RAR for urban vehicle detection," Sensors, vol. 23, no. 4, p. 1801, 2023.
14. S. Srivastava, S. Narayan, and S. Mittal, "A survey of deep learning techniques for vehicle detection from UAV images," Journal of Systems Architecture, vol. 117, p. 102152, 2021.
15. R. M. A. Latif, N. Jamal, U. Raza, H. Naeem, and J. Ahmad, "Long-Term Memory-Aware Object Tracking with Recurrent Attention Models," in International Conference on Internet of Things, Communication and Intelligent Technology, 2025, pp. 484-495: Springer.
16. S. Khan, S. Nazir, I. García-Magariño, and A. Hussain, "Deep learning-based urban big data fusion in smart cities: Towards traffic monitoring and flow-preserving fusion," Computers Electrical Engineering, vol. 89, p. 106906, 2021.
17. J. Azimjonov and A. Özmen, "A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways," Advanced Engineering Informatics, vol. 50, p. 101393, 2021.
18. N. Arora, Y. Kumar, R. Karkra, and M. Kumar, "Automatic vehicle detection system in different environment conditions using fast R-CNN," Multimedia Tools Applications, vol. 81, no. 13, pp. 18715-18735, 2022.
19. Q. Liu, Z. Li, S. Yuan, Y. Zhu, and X. Li, "Review on vehicle detection technology for unmanned ground vehicles," Sensors, vol. 21, no. 4, p. 1354, 2021.
20. Kumari, J., Ankarboina, H., Sahoo, S. K., Mishra, S. K., Raju, M. R., & Mishra, S. K. (2026). Integration of edge-enabled IoT in vehicular networks in smart cities. In Smart City Computational Paradigms (pp. 101-136). Elsevier.

21. Q. Liu, X. Wang, X. Wu, Y. Glaser, and L. He, "Crash comparison of autonomous and conventional vehicles using pre-crash scenario typology," Accident Analysis Prevention, vol. 159, p. 106281, 2021.
22. R. M. A. Latif, N. Jamal, F. Ullah, U. Raza, M. Naseer, and M. A. K. Khattak, "Secure Wireless Sensor Networks with Blockchain-Based Trust Management," in International Conference on Internet of Things, Communication and Intelligent Technology, 2025, pp. 474-483: Springer.
23. R. Corbally and A. Malekjafarian, "A data-driven approach for drive-by damage detection in bridges considering the influence of temperature change," Engineering Structures, vol. 253, p. 113783, 2022.
24. G. X. Hu, B. L. Hu, Z. Yang, L. Huang, and P. Li, "Pavement crack detection method based on deep learning models," Wireless Communications Mobile Computing, vol. 2021, no. 1, p. 5573590, 2021.
25. L. Fredianelli et al., "Traffic flow detection using camera images and machine learning methods in ITS for noise map and action plan optimization," Sensors, vol. 22, no. 5, p. 1929, 2022.
26. N. Li, Y. Zhao, Q. Pan, S. G. Kong, and J. C.-W. Chan, "Illumination-invariant road detection and tracking using LWIR polarization characteristics," ISPRS Journal of Photogrammetry Remote Sensing, vol. 180, pp. 357-369, 2021.
27. R. M. A. Latif, F. Ullah, U. Raza, M. A. Khan, L. Mostarda, and D. Cacciagrano, "Blockchain-Based Secure Communication Protocols in Healthcare IoT," in International Conference on Internet of Things, Communication and Intelligent Technology, 2025, pp. 333-344: Springer.
28. Abualhaj, M. (2024). Enhancing spam detection using hybrid of Harris Hawks and firefly optimization algorithms. Journal of Soft Computing and Data Mining, 5(2), 161-174.
29. Hussain, T., Faiz, R. B., Aljaidi, M., Khattak, A., Samara, G., Alsarhan, A., & Alazaidah, R. (2023). Maximizing test coverage for security threats using optimal test data generation. Applied Sciences, 13(14), 8252.
30. Maurya, T., Kumar, S., Rai, M., Saxena, A. K., Goel, N., & Gupta, G. (2025). Real time vehicle classification using deep learning—smart traffic management. Engineering Reports, 7(3), e70082.
31. AlJamal, M., Alquran, R., Alsarhan, A., Aljaidi, M., Al-Jamal, W. Q., & Alkoradees, A. F. (2025). Optimized novel text embedding approach for fake news detection on twitter x: Integrating social context, temporal dynamics, and enhanced interpretability. International Journal of Computational Intelligence Systems, 18(1), 22.
32. Mosleh, A., Ahmad, A., Mohammad, O., Yousef, A., Mahran, A., & Mohammad, A. (2022). A paradigm for DoS attack disclosure using machine learning techniques. International Journal of Advanced Computer Science and Applications, 13(3).
33. X. Tang, Q. Guo, M. Li, C. Wei, Z. Pan, and Y. Wang, "Performance analysis on liquid-cooled battery thermal management for electric vehicles based on machine learning," Journal of Power Sources, vol. 494, p. 229727, 2021.
34. Y. Zeng, T. Zhang, W. He, and Z. Zhang, "Yolov7-uav: An unmanned aerial vehicle image object detection algorithm based on improved yolov7," Electronics, vol. 12, no. 14, p. 3141, 2023.
35. R. M. A. Latif, F. Ullah, U. Raza, L. Mostarda, and D. Cacciagrano, "Blockchain-Enabled Ethical Governance Framework for Transparent, Accountable, and Trustworthy AI Systems: From Transparency to Ethical AI Governance," in Ethical AI Applications for Software Engineering and Development: IGI Global Scientific Publishing, 2026, pp. 93-122.
36. M. Farhan, R. M. A. Latif, F. Ullah, and M. R. Naeem, "Future trends and emerging technologies in IoT security," Edge Intelligence, pp. 209-239, 2026.
37. M.M. Abualhaj, Q.Y. Shambour, A. Alsaaidah, A. Abu-Shareha, S. Al-Khatib, M.O. Hiari, Enhancing Spam Detection Using Hybrid of Harris Hawks and Firefly Optimization Algorithms. Journal of Applied Data Sciences,5, 901-911 (2024). https://doi.org/10.47738/jads.v5i3.279
38. Mosleh M. Abualhaj, Sumaya N. Al-Khatib, Mahran Al-Zyoud, Iyas Qaddara, Mohammad O. Hiari, and Sultan Mesfer A. Aldossary, "Enhanced Network Communication Security Through Hybrid Dragonfly-Bat Feature Selection for Intrusion Detection," Journal of Communications, vol. 20, no. 5, pp. 607-618, 2025.
39. R. M. A. Latif, M. Farhan, N. A. Khan, and R. Sujatha, "Sustainable computing-based simulation of intelligent border surveillance using mobile WSN," in Navigating Cyber Threats and Cybersecurity in the Logistics Industry: IGI Global Scientific Publishing, 2024, pp. 90-122.
40. R. M. A. Latif, M. S. Obaidat, F. Ullah, and K. Mahmood, "Ai-driven energy-efficient load balancing in hybrid edge-cloud architectures for renewable energy networks," IEEE Networking Letters, 2025.