DhakaNet: Unstructured Vehicle Detection using Limited Computational Resources

Tarik Reza Toha[1], Masfiqur Rahaman[2], Saiful Islam Salim[3], Mainul Hossain[4],
Arif Mohaimin Sadri[5], and A. B. M. Alim Al Islam[6]
[1,2,3,4,6]Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
[5]University of Oklahoma, Norman, Oklahoma, USA
Email: {[1]1017052013, [2]0421052008, [3]1018052067, [4]0421052047}@grad.cse.buet.ac.bd,
[5]sadri.buet@gmail.com, and [6]alim_razi@cse.buet.ac.bd

***Abstract*—Inefficient traffic signal control system is one of the most important causes of traffic congestion in the cities of developing countries such as Bangladesh, India, Kenya, etc. This can be mitigated by adopting a decentralized traffic-responsive signal system, where vehicle detection is performed on the road through different image-based deep learning architectures amenable to limited-resource embedded platforms as available in developing countries. Deep learning architectures currently available in this regard demand high computational resources to achieve higher inference speed and better accuracy. Besides, the few existing limited-resource deep learning architectural alternatives neither attain higher inference speed nor substantial accuracy due to not overcoming the inherent limitations. To this extent, in this study, we propose a novel limited-resource deep learning architecture, namely DhakaNet, for real-time vehicle detection in on-road (street-view) traffic images. Our proposed architecture leverages enhancing Cross-Stage Partial Network and Path Aggregation Network to build the backbone and head networks, respectively. Besides, we develop a novel multi-scale attention module to extract multi-scale meaningful features from the images, where the developed multi-scale attention module boosts the detection accuracy at the cost of small overhead. Rigorous experimental evaluation of our proposed DhakaNet over three benchmark street-view traffic datasets such as DhakaAI, IITM-HeTra-A, and IITM-HeTra-B shows up to 51% faster inference speed at a similar accuracy, or up to 13% higher accuracy at a similar inference speed compared to other state-of-the-art limited-resource deep learning architectural alternatives.**

***Index Terms*—Traffic congestion, Vehicle detection, Deep learning, Computer vision, Embedded systems**

I. Introduction

Traffic congestion is one of the most severe challenges for the cities of developing countries such as Bangladesh, India, and Kenya. According to the World Bank Report [1], the average driving speed in Dhaka, i.e., the capital city of Bangladesh, is 7 kilometers per hour (kph), which is expected to be 4 kph (slower than walking speed) by 2035. Moreover, traffic congestion in Dhaka wastes about 3.2 million working hours daily and billions of dollars of the national economy annually [1]. In order to reduce traffic congestion, one of the most effective approaches is to adopt a traffic-responsive signal control system, i.e., schedule traffic signal based on current vehicular density at signalized intersections [2]. This approach requires real-time traffic density information, which can be captured through different image-based deep learning architectures [3], [4].

Most of the existing image-based learning architectures capture on-road traffic images and upload them to the cloud for necessary processing tasks such as vehicle detection [5], [6]. However, these cloud-based solutions demand high-speed network connectivity, which is not always available across all road intersections in developing countries [7], [8]. Hence, a decentralized approach needs to be adopted, where vehicle detection is performed in the embedded platforms and only the vehicle count is uploaded to the cloud. This approach alleviates the demand of high-speed network, however, imposes severe computational constraints (inherited from the embedded platforms) on the learning models. This happens as the state-of-the-art deep learning architectures demand high computational resources (GPUs) to provide faster and more accurate detection, where limited computational resources result in slower and less accurate detection [7], [9]. A few deep learning architectures such as YOLOv4-tiny [10] and YOLOv5-small [11] have recently been proposed for the resource-constrained environment, however, they cannot achieve either faster inference speed or more accurate detection in the embedded platforms.

As a remedy for the problems mentioned above, in this paper, we propose a novel deep learning architecture named *DhakaNet* for faster and more accurate vehicle detection using limited computational resources. To do so, first, we analyze the limitations of existing low-resource deep learning architectures. To overcome the limitations, we modify the existing Cross-Stage Partial Network [12] and Path Aggregation Network [13] to build our backbone and head networks respectively. Besides, we develop a novel multi-scale attention module that extracts multi-scale meaningful features from the images. This module boosts detection accuracy using a small overhead. We evaluate the performance of DhakaNet against two state-of-the-art low-resource deep learning architectures namely YOLOv4-tiny [10] and YOLOv5-small [11] over three different benchmark traffic datasets namely DhakaAI [14], IITM-HeTra-A [15], and IITM-HeTra-B [15] using an embedded system named Raspberry Pi. Based on our work, we make the following set of contributions: 1) We propose a novel low-resource deep learning architecture namely DhakaNet for faster and more accurate vehicle detection in street-view traffic images. 2) We develop a new multi-scale attention module to extract scale-aware and meaningful features from the traffic images. 3) Performance evaluation of DhakaNet against state-of-the-art low-resource architectures over three benchmark street-view traffic datasets using a Raspberry Pi confirms up to 51% faster inference speed at a similar accuracy, or up to 13% higher accuracy at a similar inference speed.

II. Related Work

Traffic density estimation is the crucial component of an automated traffic monitoring system. Over the last few years, with the help of image-based deep learning approaches, researchers achieved promising results on counting and detecting vehicles from traffic camera images, from which the traffic density information can be estimated [3]. In this regard, researchers have proposed various general object detection models in the literature to get classified vehicle count from traffic images in real-time. Now, we describe some of the object detection models chronologically.

A. Basic Object Detection Models

Ren et al., [16] proposed a real-time two-stage object detection model using region proposal networks named Faster R-CNN that was used in several traffic detection studies [8], [15]. Although this model achieves high accuracy, it requires high computational resources such as GPUs for faster inference. Hence, it cannot be used in a decentralized adaptive traffic control system. To increase inference speed, Redmon et al., [17] proposed a unified one-stage real-time object detection model named YOLO (You Only Look Once). This model applies a single neural network to the whole image, divides the image into rectangular grid regions, and predicts bounding boxes and probabilities for each region. Due to its simple architecture, it achieves much faster inference speed than Faster R-CNN. Hence, it can be used in real-time traffic detection [7]. However, the accuracy of YOLO drops slightly due to fast inference. To achieve a better trade-off between accuracy and inference speed, Liu et al., [18] proposed a single shot multibox detector named SSD that runs a convolutional network on input image only once and computes a feature map. Besides, Howard et al., [19] proposed an efficient convolutional neural network for mobile vision applications named MobileNet that requires much lower computational resources for object detection. Hence, the combination of MobileNet-SSD can be a good choice used for a decentralized adaptive traffic signal control system as suggested by Chauhan et al., [7]. However, the accuracy of such a model is not satisfactory. Next, we review some latest deep learning architectures that achieve both faster inference speed and higher accuracy.

B. Advanced Object Detection Models

Tan et al., [20] proposed a scalable and efficient object detection model named EfficientDet that improves both the accuracy and efficiency of object detection. However, it demands high computational resources such as GPUs for faster inference speed and higher accuracy. To achieve more efficiency, Bochkovskiy et al., [21] proposed YOLOv4 that achieves optimal speed and accuracy in object detection. However, it demands high-end devices similar to EfficientDet. Later, Wang et al., [10] proposed Scaled-YOLOv4 through scaling cross-stage partial network. Here, scaling the network means the modification of network depth, width, resolution, and structure. They provided both large models for high-end (GPU) devices and tiny models for low-end embedded systems. However, the YOLOv4-tiny requires a longer inference time in embedded systems due to its architectural limitations. Hence, it does not apply to a decentralized adaptive traffic control system. Similar to this scaled model, Jocher et al., [11] proposed a scalable model named YOLOv5 that can easily be adapted for both GPU-based and embedded system-based applications. Although it achieves a higher inference speed in embedded systems, its accuracy decreases due to its architectural issues.

The literature survey shows that existing deep learning architectures trade off the object detection accuracy for inference speed. Although faster inference speed is required in a resource-constrained environment, detection accuracy cannot be sacrificed to make the model applicable in real-world scenarios.

III. Proposed DhakaNet Architecture

Our DhakaNet architecture has two parts namely backbone network and head network. The backbone network extracts low-level information from raw images and forwards them to the head network. The head network learns high-level information and detects our desired vehicles from the traffic images. Figure 1 shows the block diagram of DhakaNet architecture with their inter-connectivity. Here, the modules written in red font indicate the innovative changes made over existing architectures. Next, we describe DhakaNet architecture in detail.

A. Backbone Network

We design our backbone network through modifying the structure of the existing Cross-Stage Partial Network (CSP) module. This module can extract richer gradient information using a minimal amount of computation. Figure 2a shows the block diagram of a modified CSP module (mCSP) used in DhakaNet. Here, an mCSP module contains one residual module (also known as a bottleneck), three point convolution layers having 1×1 kernels, and a feature concatenation layer. In a residual block, one point convolution layer is used before a 3×3 convolution layer to compress the feature representation that enhances the learning ability of the network [22]. Besides, the output of the residual layer is concatenated with input layers through a point convolution layer. Moreover, the concatenated output is further summarized using another point convolution layer. Thus, this module extracts rich information from the network. Note that, existing CSP module contains one or more bottleneck blocks, whereas we use one or zero bottleneck blocks in our mCSP module.

We use several mCSP modules after every down-sampling step. In Figure 2a, mCSP-32-B denotes a modified CSP module having one bottleneck and mCSP-32 denotes a similar modified CSP module without using a bottleneck. Besides, Conv-32-1-1 denotes a convolution layer having 32 filters, 1×1 kernel size, and 1 stride. We use strided convolution to down-sample the images instead of a max-pooling operation,

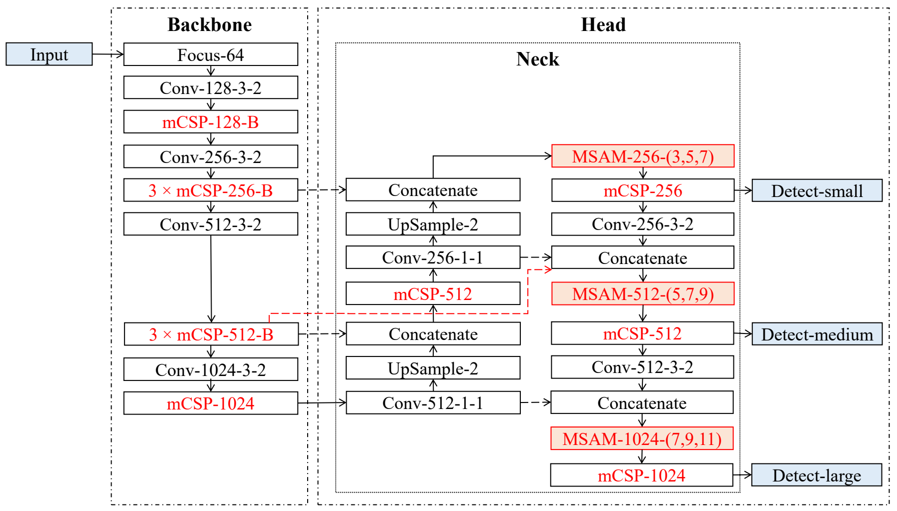
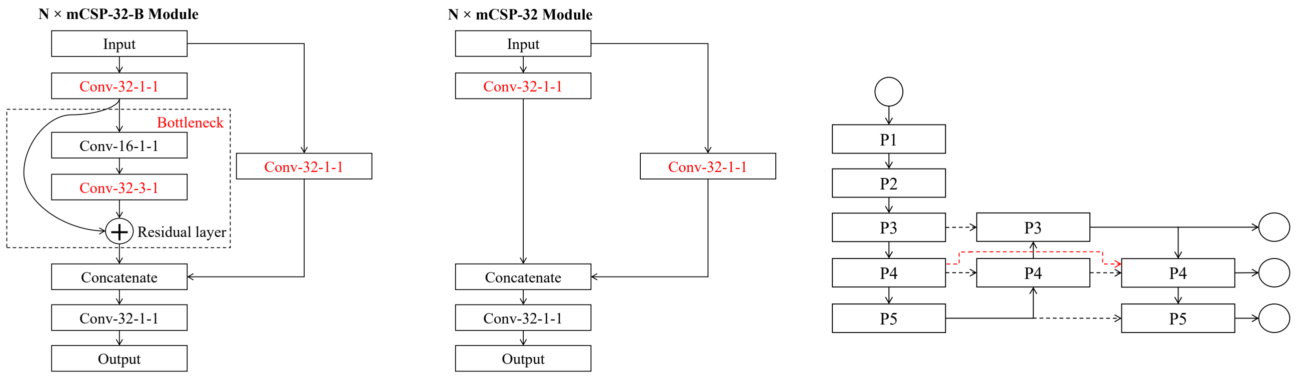


Fig. 1: Block diagram of our proposed DhakaNet architecture



(a) Modified Cross-Stage Partial Network

(b) Modified Path Aggregation Network

Fig. 2: Block diagram of modified Cross-Stage Partial Network and modified Path Aggregation Network

which increases the learning capacity of the network. At the beginning of the backbone, we use a focus layer to reduce input image resolution quickly through rearranging blocks of spatial data into depth axis followed by a convolution layer. Besides, in the last down-sampled stage, we do not use any bottleneck in the mCSP modules to reduce computation. Note that, we use batch normalization after each convolution and before activation to stabilize training, speed up convergence. Besides, we use Sigmoid Linear Unit (Swish) [23] layer as the activation function of our architecture. Next, we describe the head network.

B. Head Network

We design our head network through modifying the existing Path Aggregation Network (PANet) as the neck of DhakaNet. This network fuses the upper-level features (semantically weak) with lower-level features (semantically strong) efficiently. Our modified PANet (mPANet) adds an extra edge from the original input to the output node shown in red-colored dotted line in Figure 2b. It fuses more features with minimal overhead, which boosts the accuracy of our limited-resource architecture. To implement this mPANet, we change the internal network resolution through up-sampling or down-sampling operation and concatenate features from different levels.

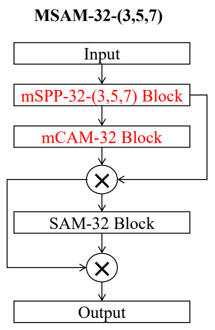


Fig. 3: Block diagram of a multi-scale attention module along with constituent blocks

After this feature fusion, we use three novel multi-scale attention modules (MSAM) and several mCSP modules. As DhakaNet utilizes a limited-resource network, it is very difficult to learn semantically rich features from the network. Hence, modifying some state-of-the-art plug-in modules, we develop the MSAM modules to increase the feature representation power of DhakaNet. Figure 3 shows the block diagram of an MSAM that contains a series of three plug-in modules such as modified spatial pyramid pooling module (mSPP), modified channel attention module (mCAM), and spatial attention module (SAM). Although we use mPANet to fuse global features across the whole network, fusing local features within the same convolution layer is necessary to improve the accuracy of the network that can be performed using the SPP blocks [24]. Besides, we exploit attention modules to emphasize meaningful features explicitly, as DhakaNet cannot learn proper feature distribution due to the small computational budget [25]. Our proposed mSPP block consists of two max-pooling layers and one average pooling layer with different kernel sizes. Note that, in the state-of-the-art approaches [11], [21], no average pooling is used in the SPP block, although average pooling is another important operation to extract distinctive features.

We use the MSAM module before each detection layer such that it can learn more discriminative information from the network. The kernel sizes of MSAM depend on feature resolution. In a higher resolution, we use smaller kernel sizes to facilitate small vehicle detection. Similarly, in lower resolution, we use larger kernel sizes for large vehicle detection. Note that, we use the anchor-based YOLOv5 detection layers [11] for our detection purpose.

IV. Training Method

We prepare the ground truth of the traffic images using conventional box labels. Next, we set the parameters of data augmentation and training configuration. Now, we describe these steps in detail.

A. Data Augmentation

We use several data augmentation techniques to improve the performance of the DhakaNet model. During the training stage, we randomly change the hue, saturation, and brightness value of the images. Besides, we use translation, horizontal flip, and mosaic [21] data augmentation techniques. Note that, we do not use rotation, vertical flip, and cutmix [21] data augmentation techniques, since they are irrelevant to our intended traffic detection problem.

B. Training Details

We use a straightforward way to train the DhakaNet architecture in an end-to-end manner. For this purpose, we use a multi-component loss function that consists of three components such as bounding box regression loss, objectness loss, and classification loss. We use Complete Intersection over Union loss (CIoU) [26] to regress the parameter of the bounding boxes. Besides, we use binary cross-entropy loss for objectness and classification losses. As we have multiple classes in the images, we use a multi-hot encoding technique to estimate the classification loss using binary cross-entropy. To optimize the loss function, we use stochastic gradient descent algorithm along with default parameters used in YOLOv5 [11]. We implement our DhakaNet architecture using PyTorch framework [11]. We use 25% of training images as the validation dataset to save the best model based on validation accuracy. Besides, we resize all images to 768×768 pixels during both training and testing stages.

TABLE I: Datasets used for performance evaluation

Attribute	DhakaAI	IITM-HeTra-A	IITM-HeTra-B
Traffic	Unstructured	Unstructured	Unstructured
Location	Dhaka, BD	Chennai, India	Chennai, India
Train-Test split	3000 : 500	1201 : 216	1201 : 216
# of object classes	21	3	4

V. Experimental Evaluation

In this section, first, we describe the experimental setup and traffic datasets used for the experimental evaluation. After that, we present the performance comparison and ablation study in detail.

A. Experimental Setup

For training purpose, we use a Ubuntu 20.04 desktop having 8 Intel Core i7-7700 CPUs (3.60 GHz) and 16 GB main memory. To accelerate the training process, we use one GeForce GTX 1070 GPU having 8 GB memory. On the other hand, we use a Raspberry Pi 4 Model B for testing purpose. Our Raspberry Pi has 4 Cortex-A72 (ARMv7) CPUs (1.5 GHz) and 4 GB main memory. Note that, we do not use any embedded GPU in our Raspberry Pi. Hence, it serves as a limited-resource computing system for our experimental evaluation.

B. Traffic Datasets

We evaluate our proposed DhakaNet architecture on three traffic datasets namely DhakaAI [14], IITM-HeTra-A [15], and IITM-HeTra-B [15]. DhakaAI dataset contains non-lane-based heterogeneous road traffic images of Dhaka city. As the vehicles in Dhaka do not follow lane discipline, the images of DhakaAI contain severe occlusion that makes the detection more challenging. Besides, the dataset has images from both day and night. This dataset has 3000 training images and 500 test images. Besides, it has 21 different classes of vehicles containing both motorized and human-powered vehicles. Note that, there is no ground-truth available for DhakaAI test images, hence, we manually label these images.

IITM-HeTra datasets contain similar unstructured traffic images of Chennai city. Both of them have 1201 training and 216 testing images. IITM-HeTra-A has 3 different classes of vehicles and IITM-HeTra-B has 4 different classes of vehicles. Note that, all vehicles of IITM-HeTra images are motorized. We summarize the demography of the aforementioned datasets in Table I.

C. Evaluation Results

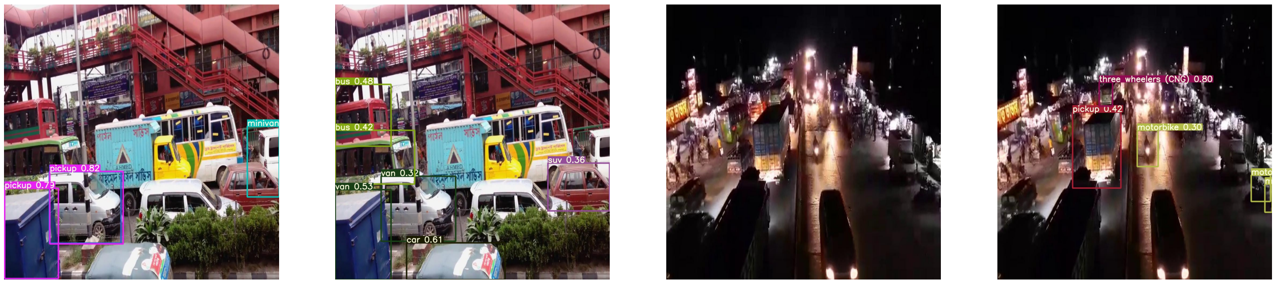
We compare the performance of DhakaNet against two recent limited-resource deep learning architectures such as YOLOv4-tiny [10] and YOLOv5-small [11] on three traffic datasets. DhakaNet outperforms both of them in terms of either faster inference speed, or higher accuracy. Note that, we average the results over five iterations for a particular training configuration in the performance evaluation, as the neural network possesses stochastic nature by default. Besides,

TABLE II: Performance comparison of DhakaNet against YOLOv4-tiny [10] and YOLOv5-small [11] over DhakaAI [14] and IITM-HeTra datasets [15]

Dataset	Model	Scaling factor	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Inference time (s)	FPS
DhakaAI [14]	YOLOv4-tiny [10]	N/A	16.2	7.2	27.8	0.04
	YOLOv5-small [11]	0.50	8.7 ($\pm$0.3)	4.3 ($\pm$0.2)	7.4	0.14
	DhakaNet	0.29	9.9 ($\pm$0.3)	5.0 ($\pm$0.1)	6.7	0.15
	DhakaNet-scaled	0.23	8.9 ($\pm$0.2)	4.5 ($\pm$0.1)	4.9	0.20
IITM-HeTra-A [15]	YOLOv4-tiny [10]	N/A	95.9	47.2	30.0	0.03
	YOLOv5-small [11]	0.50	94.6 ($\pm$1.6)	50.8 ($\pm$1.4)	5.6	0.18
	DhakaNet	0.29	95.8 ($\pm$1.3)	52.2 ($\pm$1.0)	5.0	0.20
	DhakaNet-scaled	0.23	95.5 ($\pm$0.6)	51.9 ($\pm$1.1)	3.7	0.27
IITM-HeTra-B [15]	YOLOv4-tiny [10]	N/A	95.4	48.2	29.9	0.03
	YOLOv5-small [11]	0.50	94.7 ($\pm$1.2)	50.8 ($\pm$1.4)	5.6	0.18
	DhakaNet	0.29	95.4 ($\pm$0.2)	52.5 ($\pm$0.7)	5.0	0.20
	DhakaNet-scaled	0.23	95.6 ($\pm$0.9)	52.5 ($\pm$0.4)	3.7	0.27

TABLE III: Ablation studies of DhakaNet over DhakaAI dataset [14]

Model	mCSP	mPANet	MSAM	Scaling factor	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Inference time (s)	FPS
DhakaNet-α	No	No	No	0.29	7.6 ($\pm$0.5)	3.8 ($\pm$0.4)	3.6	0.28
DhakaNet-β	Yes	No	No	0.29	8.3 ($\pm$0.5)	4.1 ($\pm$0.2)	5.8	0.17
DhakaNet-γ	Yes	Yes	No	0.29	8.8 ($\pm$0.7)	4.3 ($\pm$0.5)	6.0	0.17
DhakaNet-final	Yes	Yes	Yes	0.29	9.9 ($\pm$0.3)	5.0 ($\pm$0.1)	6.7	0.15



(a) Out# 1: YOLOv5-small (b) Out# 1: DhakaNet (c) Out# 2: YOLOv5-small (d) Out# 2: DhakaNet

Fig. 4: Qualitative analysis between YOLOv5-small [11] and DhakaNet over two challenging test images in DhakaAI dataset

we train all models from the scratch for a fair comparison. Moreover, we set our confidence threshold to 0.3, i.e., the detection having lower than 30% confidence score gets ignored. Next, we present the performance evaluation results over three different datasets.

1) DhakaAI Dataset: Table II shows the performance evaluation results of DhakaNet against YOLOv4-tiny [10] and YOLOv5-small [11] over DhakaAI dataset [14]. Here, we can see that YOLOv4-tiny requires 27.8 seconds for detecting vehicles per image on average, which is not a realistic solution for a decentralized adaptive traffic control system. Our proposed DhakaNet achieves 13% higher accuracy at a similar inference speed compared to YOLOv5-small. Besides, the down-scaled version of DhakaNet achieves 50% faster inference speed at a similar accuracy. Hence, DhakaNet outperforms the existing YOLOv5-small in terms of either faster inference speed, or higher accuracy.

2) IITM-HeTra Datasets: We show the performance evaluation results of DhakaNet on IITM-HeTra datasets [15] in Table II. For both datasets, we can see that YOLOv4-tiny requires a much longer inference time on average, which is not a pragmatic solution as described before. On the other hand, our proposed DhakaNet and DhakaNet-scaled achieve 51% faster inference speed and similar accuracy on both IITM-HeTra datasets compared to YOLOv5-small.

3) Ablation Studies: To analyze the effectiveness of our proposed modules of DhakaNet, we conduct an ablation study of DhakaNet over DhakaAI dataset [14]. For this purpose, we turn on mCSP, mPANet, and MSAM modules sequentially and evaluate corresponding accuracy and speed. The ablation results are delineated in Table III. Here, we can see that DhakaNet having all the three modules achieves the highest accuracy among all the other alternative variants. Note that, the speed is getting dropped because of adding new modules to DhakaNet architecture.

D. Qualitative Analysis

Figure 4 shows the detections of YOLOv5-small [11] and DhakaNet over two challenging test images of DhakaAI dataset. For severe occlusion, DhakaNet performs much better than YOLOv5-small. Besides, in the night scene, YOLOv5-small misses all vehicles whereas DhakaNet detects several vehicles.

VI. Conclusion and Future Work

Decentralized traffic-responsive signal system is one of the possible solutions to traffic congestion in the cities of developing countries. It requires on-road vehicle detection using limited-resource image-based deep learning architectures. However, existing low-resource architectures exhibit either low inference speed or low detection accuracy due to their inherent limitations. Hence, we propose a new architecture named DhakaNet for faster and more accurate vehicle detection using embedded systems through enhancing Cross-Stage Partial Network and Path Aggregation Network as well as adding several novel multi-scale attention modules. We evaluate our proposed DhakaNet against two state-of-the-art limited-resource deep learning architectures over three unstructured traffic datasets such as DhakaAI, IITM-HeTra-A, and IITM-HeTra-B on a Raspberry Pi. DhakaNet confirms up to 50% faster inference speed at a similar accuracy, or up to 13% higher accuracy at a similar inference speed compared to the other state-of-the-art approaches. In the future, we plan to integrate DhakaNet in a real traffic signaling module and deploy the complete system in real-world.

Acknowledgment

This research work was conducted under a fellowship from the ICT Division, Government of the People's Republic of Bangladesh. Besides, the Department of CSE, BUET is thankfully acknowledged for providing the GPU and other necessary support for this research work.

References

[1] J. Bird, Y. Li, H. Z. Rahman, M. Rama, and A. J. Venables, *Toward Great Dhaka : A New Urban Development Paradigm Eastward.* Washington, D.C., USA: The World Bank, 2018. [Online]. Available: https://doi.org/10.1596/978-1-4648-1238-5

[2] Y. Wang, X. Yang, H. Liang, and Y. Liu, "A Review of the Self-Adaptive Traffic Signal Control System Based on Future Traffic Environment," *Journal of Advanced Transportation*, vol. 2018, June 2018. [Online]. Available: https://doi.org/10.1155/2018/1096123

[3] J. Chung and K. Sohn, "Image-Based Learning to Measure Traffic Density Using a Deep Convolutional Neural Network," *Transactions on Intelligent Transportation Systems (ITS)*, vol. 19, no. 5, pp. 1670–1675, May 2018. [Online]. Available: https://doi.org/10.1109/TITS.2017.2732029

[4] D. Biswas, H. Su, C. Wang, A. Stevanovic, and W. Wang, "An Automatic Traffic Density Estimation using Single Shot Detection (SSD) and MobileNet-SSD," *Physics and Chemistry of the Earth, Parts A/B/C*, vol. 110, pp. 176–184, April 2019. [Online]. Available: https://doi.org/10.1016/j.pce.2018.12.001

[5] W.-H. Lee and C.-Y. Chiu, "Design and Implementation of a Smart Traffic Signal Control System for Smart City Applications," *Sensors*, vol. 20, no. 2, January 2020. [Online]. Available: https://doi.org/10.3390/s20020508

[6] J. Nubert, N. G. Truong, A. Lim, H. I. Tanujaya, L. Lim, and M. A. Vu, "Traffic Density Estimation using a Convolutional Neural Network," National University of Singapore, Tech. Rep., 2018. [Online]. Available: https://arxiv.org/abs/1809.01564

[7] M. S. Chauhan, A. Singh, M. Khemka, A. Prateek, and R. Sen, "Embedded CNN Based Vehicle Classification and Counting in Non-Laned Road Traffic," in *Proceedings of the 10th International Conference on Information and Communication Technologies and Development (ICTD).* Ahmedabad, India: ACM, January 2019, pp. 1–11. [Online]. Available: https://doi.org/10.1145/3287098.3287118

[8] C. Yeshwanth, P. S. A. Sooraj, V. Sudhakaran, and V. Raveendran, "Estimation of Intersection Traffic Density on Decentralized Architectures with Deep Networks," in *Proceedings of the 3rd International Smart Cities Conference (ISC2).* Wuxi, China: IEEE, September 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISC2.2017.8090799

[9] Q. Mao, H. Sun, Y. Liu, and R. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," *IEEE Access*, vol. 7, pp. 133 529–133 538, September 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2941547

[10] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling Cross Stage Partial Network," in *Proceedings of the 34th International Conference on Computer Vision and Pattern Recognition (CVPR).* Nashville, TN, USA: IEEE, June 2021, pp. 13 029–13 038. [Online]. Available: https://arxiv.org/abs/2011.08036

[11] G. Jocher *et al.*, "YOLOv5: Ultralytics LLC," https://github.com/ultralytics/yolov5, last accessed on June 11, 2021.

[12] C. Wang, H. Mark Liao, Y. Wu, P. Chen, J. Hsieh, and I. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *Proceedings of the 33th International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).* Seattle, WA, USA: IEEE, June 2020, pp. 1571–1580. [Online]. Available: https://doi.org/10.1109/CVPRW50498.2020.00203

[13] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *Proceedings of the 31st International Conference on Computer Vision and Pattern Recognition (CVPR).* Salt Lake City, UT, USA: IEEE, June 2018, pp. 8759–8768. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00913

[14] A. Shihavuddin and M. R. A. Rashid, "DhakaAI 2020," https://doi.org/10.7910/DVN/POREXF, last accessed on June 11, 2021.

[15] D. Mittal, A. Reddy, G. Ramadurai, K. Mitra, and B. Ravindran, "Training a Deep Learning Architecture for Vehicle Detection Using Limited Heterogeneous Traffic Data," in *Proceedings of the 10th International Conference on Communication Systems Networks (COMSNETS).* Bengaluru, India: IEEE, January 2018, pp. 589–294. [Online]. Available: https://doi.org/10.1109/COMSNETS.2018.8328279

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 6, pp. 1137–1149, June 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the 29th International Conference on Computer Vision and Pattern Recognition (CVPR).* Seattle, WA, USA: IEEE, June 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proceedings of the 14th European Conference on Computer Vision (ECCV).* Amsterdam, Netherlands: Springer, September 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0_2

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv, Tech. Rep. 1704.04861, April 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[20] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *Proceedings of the 33th International Conference on Computer Vision and Pattern Recognition (CVPR).* Seattle, WA, USA: IEEE, June 2020, pp. 10 778–10 787. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.01079

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv, Tech. Rep. 2004.10934, April 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[22] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the 30th Conference on Computer Vision and Pattern Recognition (CVPR).* Honolulu, HI, USA: IEEE, July 2017, pp. 6517–6525. [Online]. Available: https://doi.org/10.1109/CVPR.2017.690

[23] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a Self-Gated Activation Function," arXiv, Tech. Rep. 1710.05941, October 2017. [Online]. Available: https://arxiv.org/abs/1710.05941v1

[24] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection," *Information Sciences*, vol. 522, pp. 241–258, June 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2020.02.067

[25] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun, "ThunderNet: Towards Real-Time Generic Object Detection on Mobile Devices," in *Proceedings of the 17th International Conference on Computer Vision (ICCV).* Seoul, Korea (South): IEEE, November 2019, pp. 6717–6726. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00682

[26] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation," arXiv, Tech. Rep. 2005.03572, September 2020. [Online]. Available: https://arxiv.org/abs/2005.03572