VEHICLE ANALYTICS FOR URBAN ROAD POLICY-MAKING USING STATE-OF-THE-ART DEEP LEARNING NETWORKS

Research Paper

Nihar Ranjan Behera, Swiss School of Business and Management, nihar1773@gmail.com

Mario Silic, Swiss School of Business and Management, mario@ssbm.ch

College of Environmental Health and Safety, Croatia

Abstract

Vehicle counting is very essential for the urban policymaker especially for traffic signal management, infrastructure development, and assessing the need of various road facilities. Vehicle counting through Deep learning on road images is very promising and can provide various others information like traffic categories, speed, etc. We have used two state-of-the-art deep learning-based vehicle detection frameworks including YOLOv4 and Faster-RCNN with the use of ResBlock, we have modified the original YOLOv4 to get real- time vehicle count from traffic stream, while Inception-ResNet- v2 is used as the backbone of the Faster-RCNN to get the highly accurate count but offline. Both models are trained on the UA- DETRAC dataset. Modified-YOLOv4 has achieved around 89% accuracy, while Faster-RCNN has around 95%.

Keywords: traffic signal management, deep learning, road images

1 Introduction

Vehicle counting from a traffic video stream is a very essential part of the modern-day transport system, such vehicle analytics offers valuable insight regarding the traffic flow which can help policymakers to manage the traffic efficiently. In the urban environment, traffic congestion is a growing issue that leads to wastage of long working hours, consumption of extra fuel that leads to carbon emissions, delay in food and business deliveries, and impacts various other social, environmental, and economic aspects [1]. Traffic flow analytics at a traffic interaction is also used to make the traffic control robust through dynamic signal timing [2] that can manage the traffic flow efficiently and eventually bring significant economic benefits through the reduction of fuel usage and lesser carbon emission.

Conventional vehicle analytics approaches use various special-purpose sensors such as microwave, magnetic coil, or ultrasonic detectors are the main component for vehicle counting. Although these sensors are very good at counting performance, higher installation costs and lack of various other information about the traffic flow such as traffic density, vehicle speed, and vehicle types. Vehicle detection is the first and core step in obtaining the traffic flow information and eventually take decisions to manage the traffic and other policy-making measures [3]. Object detection includes classification and localization of the objects from an image, and object counting is performed based on the detected bounding boxes. Traditional object detection involves motion and handcrafted features to detect vehicles directly from an image. These approaches usually relied on handcrafted features such as HOG [4], Haar-like [5], and SIFT [6], then these extracted features are feed into the classifier, such as Support Vector Machine (SVM), Decision Tree, Adaboost, etc. Deformable model for object detection using built that trained SVM classifier through HOG features. This model performs better-provided object has some deformation or various in scale, but is unable to perform good enough in the

presence of rotations lacks stability, and is pretty slow in detection.
In recent times, Convolutional Neural Networks (CNNs) have obtained fabulous performance in object classification, segmentation, and detection. Deep learning-based object detection frameworks have given a huge boost to accuracy compared with the conventional detection methods. However, vehicle detection in the complex environment when vehicles need to detect on the road intersection. Region-based object detection framework, RCNN, used selective search to extract all the potential regions, this method was very slow [7]. Fast- RCNN included bounding box regressor and multi-task loss function [8]. Faster-RCNN has greatly improved the accuracy and efficiency compared to previous region-based object detection methods, and remains state of the art as of now, but pretty slow and hard to optimize due to complex pipeline [9]. YOLO [10] framework is an end-to-end object detector having a single pipeline that makes it very efficient and simple to optimize. YOLOv3 [11] and YOLOv4 [12] are state-of-the- art in the YOLO family. YOLOv4 is designed in the context of deployment that also provides flexibility in the backbone and head part of the detection pipeline. This paper used two state-of-the-art detection frameworks Faster-RCNN and YOLOv4 and proposed some changes to make them more accurate and efficient. This is for two use cases, 1st is to obtain a highly great count with Faster-RCNN but offline because it is not for real-time detection, and 2nd is to get a real-time vehicle count with good accuracy but less as compared to Faster-RCNN. After detection, vehicles are counted by defining a reference point in the traffic stream. We have made the reference point by making a line on the streaming video, and a count will be made when a detected vehicle's bounding box will touch the reference line.

Our main contributions are as follows:

1. Real-time vehicle counting the inclusion of ResBlock-D modules in the YOLOv4
2. Accuracy improvement in the Faster-RCNN through the use of Inception-Resnet-v2 network.

In the next section, we have described the different proposed approaches for object detection. Relating to the problem in previous approaches, we have explained our object detection in 3rd section. In the 4th section, we have elaborated on the dataset, training setup, and evolution of both object detectors. Lastly, the 5th section summarizes our work conclusion and feeds the future course.

2 Literature review

From general to domain-specific object detection, deep learning models have been immensely adopted in the whole field of Computer Vision. For the feature extraction from the image, the backbone of advanced object detection networks is based on Deep learning [13]. Conventional Neural Network (CNN) is the baseline network for building the different deep learning models [14]. Layers of CNN act as a feature extractor and generate feature maps as output where feature maps are summarized representative of different features of objects. The output of the input layer is a matrix containing intensity values of different colour channels (e.g., RGB). The feature map of different internal layers is a multichannel version of the input image, where each "pixel" refers to a specific feature. Neurons form a connected network where a neuron in the successive layer relates to multiple neurons of the preceding layer. Transformational operations [15], [16], [17] are used to build feature maps, and commonly used transformations are convolution and pooling. Convolution (filtering) operation involves linear multiplications of a filter matrix with the values of a receptive field of the feature map and introduces nonlinearity (like as sigmoid [18]. To make the single value representative of different features, different Pooling techniques, such as average pooling, max pooling, and local contrast normalization [19] which summarizes the output of the feature map.

Object detection involves localization and classification of objects where localization refers to show the presence of the object through bounding box and classification aims to assign category labels against each detected object. The object detection approaches are broadly coupled into two types. One is based on generating region proposals and classification of generated proposals, while the other type takes detection as a regression task where a single unified network is responsible for localization and classification. The region proposal-based methods mainly include R-CNN [7], spatial pyramid pooling (SPP)-net [20], Fast R-CNN [8], Faster R-CNN [9], region- based fully convolutional, Mask R-CNN

[17], network (R- FCN) [21], and feature pyramid networks [16]. The methods based on regression/classification mainly include AttentionNet [22], MultiBox [23], G-CNN [24], but YOLO [10], Single Shot MultiBox Detector (SSD) [22], and YOLOv3 [11] are state-of-the-art and widely used methods.

Faster-RCNN is selected due to its accuracy, and this is accuracy is due to Deep learning-based region proposal network called RPN. This kind of state-of-the region proposal network is missing another detector. Although, SSD is a single pipeline model, but it has issues with scale variability of the objects, whereas YOLOv4 takes the anchor boxes through clustering and perform very fast due to single YOLOv3 for vehicle detection which mitigates the false positives during autonomous driving which can end up in fatal accidents and hinder safe and efficient driving. Based on he region proposal concept, Hu et al [27] proposed a Faster R-CNN method for vehicle detection and Long Short-term Memory networks for more accurate long-term motion extrapolation. Cao et al [28] have used the Single-shot multi- box detector which combines the anchor mechanism and feature pyramid structure of the Faster R-CNN with the regression idea of YOLO and formulates a single end-to-end pipeline for the detection of multiple objects. The proposed model is capable of drawing multiple bounding boxes with classification labels.

3 Methodology

3.1 Yolov4 Base Vehicle Detection

YOLOv4-tiny [29] object detector is a light version of YOLOv4 [12] which enhances the detection speed. With this light version, YOLOv4 can attain around 370 frames per second (FPS) with very good accuracy on a GPU-enabled machine having 1080Ti GPU. The YOLOv4-tiny includes Cross-stage-Partial-connections (CSP) Darket53-tiny as a backbone feature extractor instead of CSPDarket53 that was used in the original YOLOv4. YOLOv4-tiny network uses cross-stage partial block as a residual block which enhances the accuracy but increases the model complexity and eventually decreases the FPS rate. To proceed with object detection in real-time on embedded devices with a better accuracy tradeoff, an improved version of YOLOv4-tiny is proposed.

To enhance the processing speed, we have used the Residual block (ResBlock instead of two CSPBlock as in YOLOv4-tiny [1]. ResBlock unit uses two direct paths network handle input representation map. In this two-path network, the path T network has three 1 x 1 and 3 x 3 convolutional (Conv) layers with stride 2 followed by another 1 x1 Conv layer. Another network, Path B, has two 3 x 3 max Pooling with stride 3 followed by 1 x Conv layer. Compared with CSPBlock used on the original YOLOv4-tiny [29], our ResBlock (Fig. 1) removes the first 3 x 3 Conv in CSPBlock, and replaces the consequent 3 x 3 Conv layers with 1 x 1 Conv layers in the Path T network to make the detection network efficient. The proposed ResBlock unit adds pooling and Conv in the Path B network, but this extra computation overhead is very small as compared to reduce in computation in the Path T network. The floating-point operations (FLOPs) are analyzed to determine the computational complexity of the CSPBlock and proposed ResBlock. FLOPs can be described as follow:

$$FLOPs = \sum_{l=1}^{N} M_l^2 . F_l^2 . C_{l-1} . C_l \qquad (1)$$

pipeline.
To achieve higher accuracy, region proposal networks like Faster R-CNN [9] are used, while regression/classification methods are impressive at efficiency. Higher accuracy with real-time detection is very critical for safety and real-time control in driver assistance systems and autonomous vehicles. YOLOv4 has come up with a great equilibrium between accuracy and efficiency and seemed a good fit for real-time scenarios.

3.2 Vehicle Detection

Recent work using CNNs is very effective for vehicle detection on-road scenarios [25]. Choi et al [26] have used Here S is the sum of all the Conv layers, ⟦ M_l ⟧ ^2 is the output feature vector of the corresponding lth layer, ⟦ F_l ⟧ ^2 is the filter size, while C_l and C_{l-1} refer to output and input channel count, respectively. For comparison, suppose an input
of 224x 244 with 64 channels, and using (1), FLOPs of ResBlock used in the proposed detection model:

$$FLOPs = 104^2 * 1^2 * 64 * 32 + 52^2 * 3^2 * 32^2 + 52^2 * 1^2 * 32 * 64 + 64 * 52^2 * 2^2 + 52^2 * 1^2 * 64^2 \quad (2)$$

$$FLOPs = 6.4\,x\,10^7 \quad (3)$$

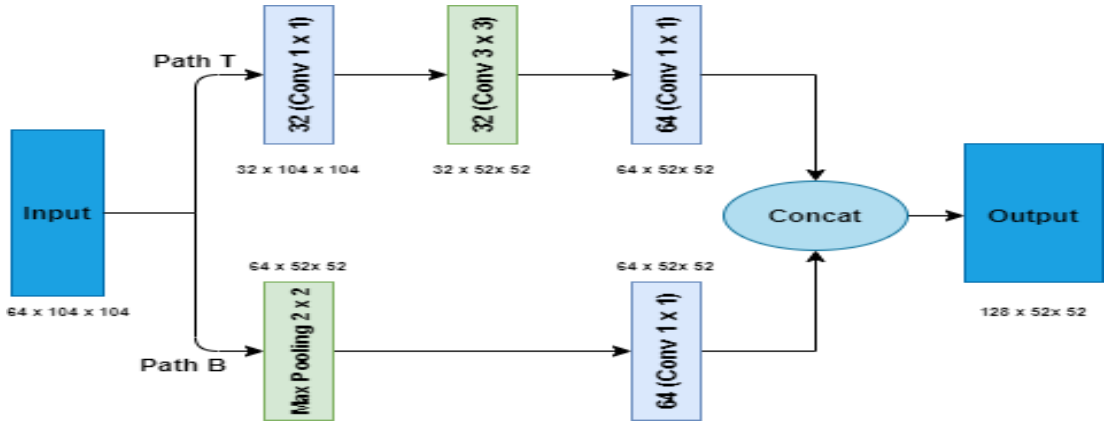

Figure 1.ResBlock-D Modules

The FLOPs of CSPBlock used in YOLOv4-tiny [1] against the same image:

$$FLOPs = 104^2 * 3^2 * 64^2 + 104^2 * 3^2 * 64 * 32 + 104^2 * 3^2 * 32^2 + 104^2 * 1^2 * 64^2 \quad (4)$$

$$FLOPs = 7.4\,x\,10^8 \quad (5)$$

From (2) and (3), we can determine that 1:10 is the computation in terms of FLOPs in ResBlock and CSPBlock. FLOPs comparison shows that ResBlock is much less complex than CSPBlock.

Although the inclusion of ResBlock in the YOLOv4-tiny detector makes it much faster than CSPBlock, however, it affects the object detection accuracy. To get a better tradeoff between efficiency and accuracy, two auxiliary residual blocks are also built and included in the ResBlock unit. The proposed backbone network is shown in figure 2.

The output representation of ResBlock is fused with a shallow representation of the backbone model through element-wise sum operation, and this fused representation is used as input to successive layers of the backbone model. The fusing process of representation of ResBlock and backbone model can be expressed as:

$$O^i = f^i(O^{i-1}) + Or^i \quad (6)$$

Here i is the index of the layers, $f^i()$ is the fusion function between the input and output in the ith layer network, O^{i-1} refers to i-1th layer's output and i th layer's input, and Or^i is the output of the proposed ResBlock. This fusion catalyzes the convergence between deep and shallow networks. With the fusion mechanism, the network tends to learn more information to enhance the accuracy, while preventing the large step-size increase of calculation.

In the backbone of YOLOv4-tiny [29], the Residual network module uses 3 x 3 filters for feature extraction.

Although 3 x 3 receptive fields can extract more localize information while losing global contextual information and eventually reduces the detection accuracy. We have compensated this loss of global representations by using two consecutive 3 x Conv layers to get the receptive field of size 5 x 5 in the auxiliary ResBlock. This auxiliary model passes on the obtained global representation to the backbone network. Then backbone network joins the local contextual information extracted from the smaller (3 x 3) receptive field and global representation extracted from the bigger (5 x 5) receptive field that gives extra information about the object. This combining of global and local information not only enhances the network depth but also, advances the semantic information. The attention mechanism can process and transmit the crucial feature and eliminate the invalid features through channel suppression. To extract more effective feature representations, we have introduced spatial and channel attention modules in the auxiliary network. The channel attention module emphasizes the interpretation of the informative part of the given input image and sees what is meaning in it, whereas, the spatial attention module emphasizes the spatial location of the informative part of the input, supportive to the channel attention. To simultaneously realize spatial and channel attention, we have used the Convolutional Block Attention Module (CBAM) [3]. The used CBAM can be described as:

$$F^c = M^c(F^i) \odot F^i \qquad (7)$$

$$F^s = M^s(F^c) \odot F^c \qquad (8)$$

Here $F^i \in R^{C \times H \times W}$ is the input feature map, "Θ" refers to element-wise multiple, F^c and F^s are the output feature maps, M^c and M^s are the channel and spatial attention functions, respectively. The channel attention function $M^c(F)$ and spatial attention function $M^s(F)$ are expressed as:

$$M^c(F^i) = S(MLP(avgPooling(F^i)) + MLP(maxPooling(F^i)) \qquad (9)$$

$$M^s(F^c) = C^{5\times 5}[maxPooling(F^c); +avgPooling(F^c)] \qquad (10)$$

Here $S()$ is the sigmoid function, MLP () is the multi-layer perceptron, and $C^{7\times 7}$ is the convolutional operation having a filter size of 5 x 5. Max Pooling and average pooling operations in spatial attention function are combined through concatenation which is referred to as ':'.

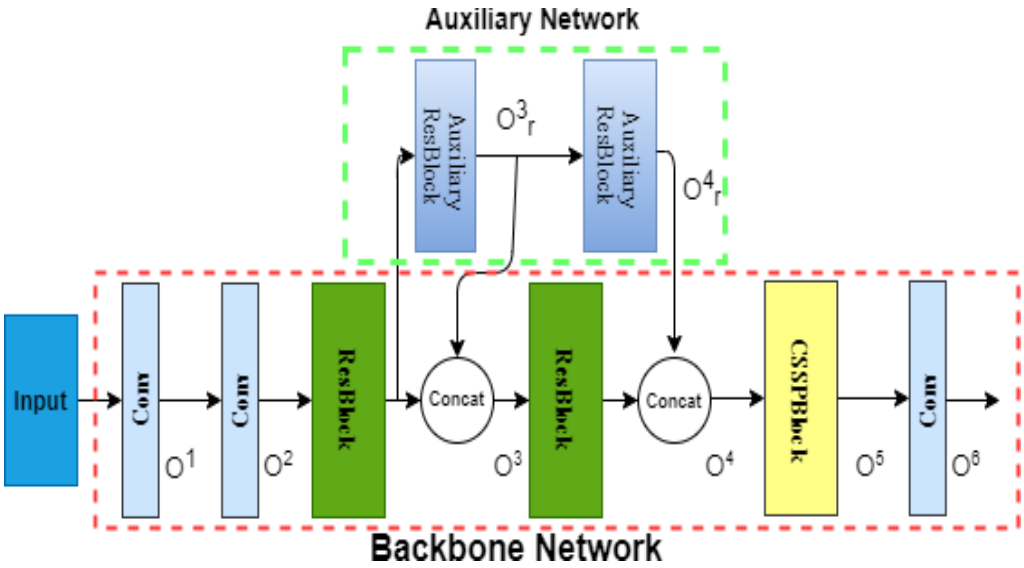

Figure 2.Proposed backbone network

Fig. 3 shows the proposed auxiliary network having two Conv layers to obtain the global contextual information, and channel and spatial attention to getting more effective information. Output representation of first Conv layer output received from spatial attention operating are concatenated to combine both outputs which is the output of the auxiliary network. Then the final output of the auxiliary network is combined with the output of the Residual network of the backbone network and used as input of the next residual network there. This joining of both outputs enhances the backbone network to extract local and global information of the object and increases the accuracy of the detection network.

The architecture of the whole YOLOv4-tiny object detector is shown in Fig. 4 where the proposed network is distinguished with blue color. Compared to YOLOv4-tiny [29], the proposed object detector has replaced both CSPBlock units with two ResBlock. Moreover, the auxiliary network is also designed by using two 3 x 3 Conv layers, channel attention module spatial attention module, and concatenation operation to obtained global information. Finally, auxiliary and backbone networks are combined to make a feature extractor.

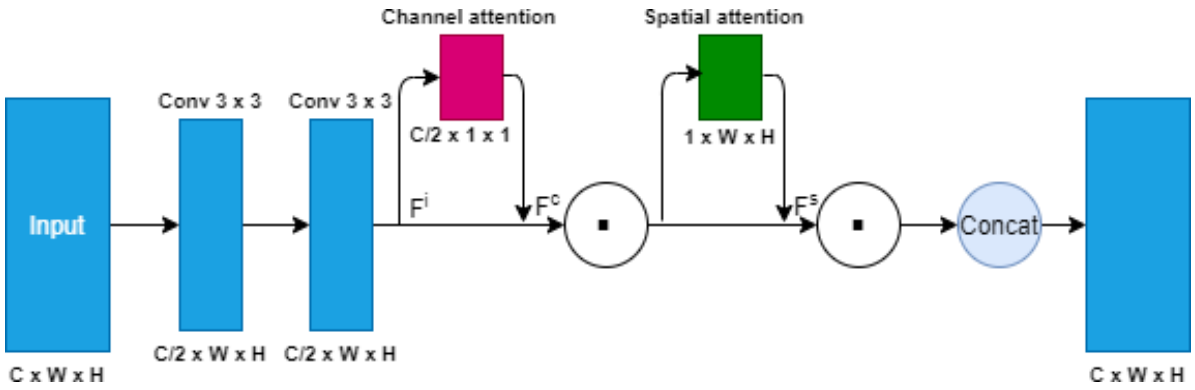

Figure 3.Auxiliary Residual Network

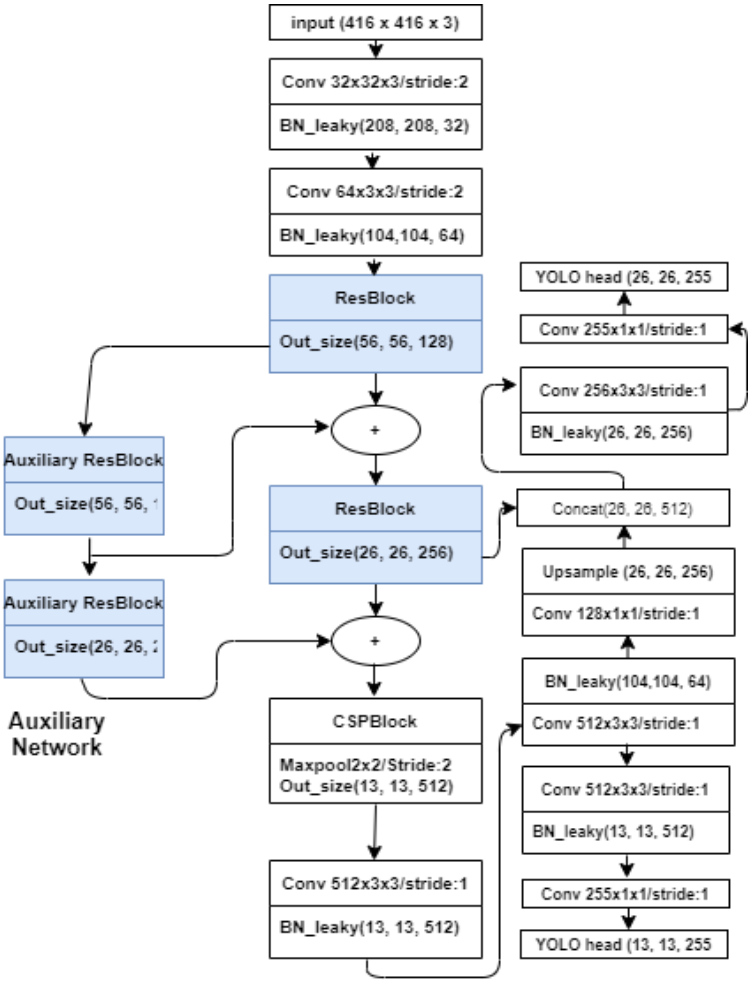

Figure 4.YOLOv4-tiny architecture with proposed changes in blue colour [36]

3.3 Faster-RCNN based vehicle detection

Faster-RCNN [12] uses region proposal algorithms to generate potential objects containing sub-part of the image. Fig.5 shows the detection pipeline of the Faster-RCNN. It consists of a region proposal network (RPN) and a Faster- RCNN detector head to classify the proposed regions. RPN is used to obtain the object locations. RPN shared layers with the detector specially to extract the region proposal by injecting a small network on the top of the shared conventional of the feature map. This added network extracts lower-dimensional representation from the representation map. In the end, the classification head estimates the objectness in the proposed regions and the regression head to predict the bounding box coordinates.

To enhance the performance of Faster-RCNN, two kinds of approaches are used. 1st approach is to increase the horizontal and vertical length of the original feature extractor without altering the base architecture. This approach is required to have more and more as the network becomes deeper and deeper. 2nd method is to change the architecture of the backbone of the detection pipeline using state-of-the-art feature extraction networks such as ResNet, Inception, and VGG, etc.). This approach improves the detection performance more as compared to the first and more favorable because of the performance of the recent classification network.

Inception-Resnet-v2 [30] is a combined form of Inception [31] and Residual network [32]. It replaces the concatenation block in the Inception with the Residual block. This improves the detection performance of the Faster-RCNN due to residual connections without compromising the efficiency of the network. Fig. 6 shows the standard layer structure of an Inception-Resnet-v2 network. This feature extract has shown promising results in the ILSVRC-2012 validation set. The use of inception extracts multi-scale features that help to detect the moving vehicles.

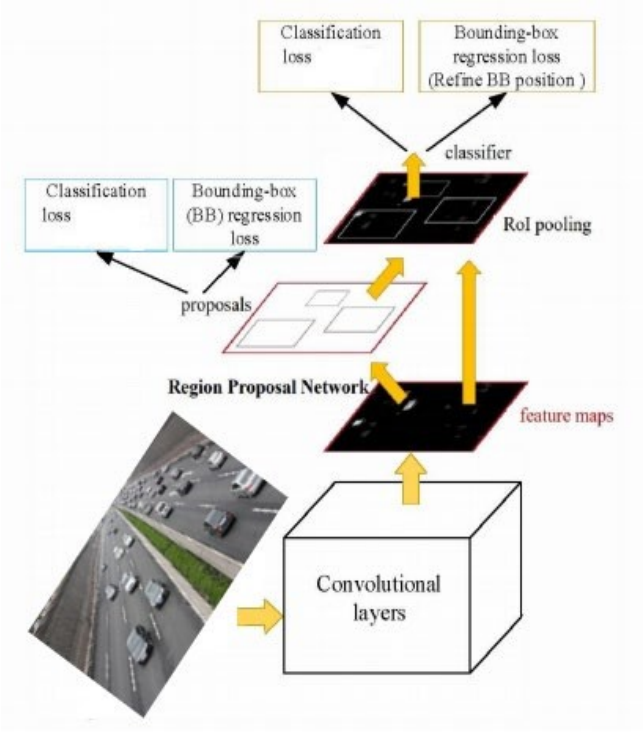

Figure 5.Faster-RCNN model for vehicle detection [9]

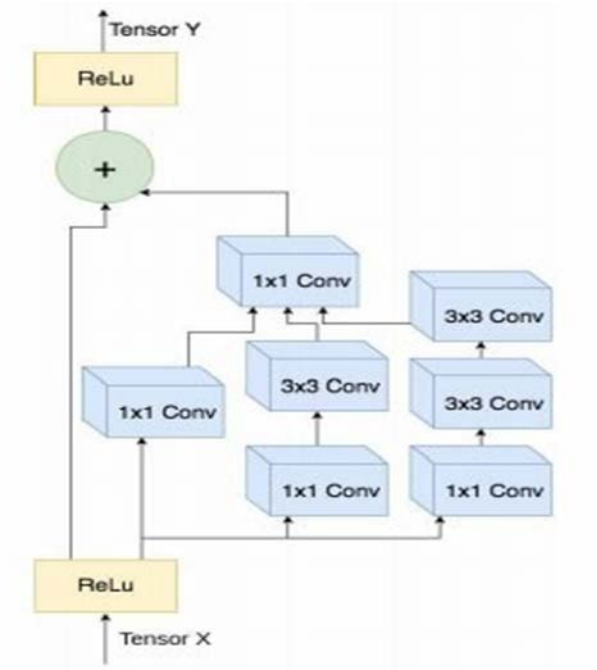

Figure 6.The standard structure of the Inception-Resnet-v2 layer [30]

Inspired by [12], we have followed the same strategy to train our Faster-RCNN model. Region proposal network (RPN) is trained separately, and backbone CNN was find- tunned on the MS COCO [33] dataset for the region proposal task. To handle the scale and aspect ratio variations, anchors of the bounding box are used in the region proposal network, and when the Intersection of Union (IoU) and rate of overlap between the ground truth box and the anchor box is greater than 0.75, the anchor box set to be positive and kept as negative in case of overlapping rate is less than 0.25. The ratio between the number of positive and negative samples is kept at 1:1, which will reduce the number of negative samples. Faster-RCNN is trained on the region proposed by the RPN. Faster-RCNN is pre-trained on MS COCO, then by fixing the shared layer of the Faster-RCNN and RPN, unique layers of the RPN are fine-tuned with Faster-RCNN. Then, unique layers of the Faster-RCNN are fine-tuned by keeping the shared layers fixed. Training of the deep learning model like Faster-RCNN requires to have millions of images, so we have used transfer learning to fine-tune the model on a specialized task like vehicle counting in our case. So, the network is trained on the MS COCO dataset and fine-tunned on the UA-DETRAC dataset.

4 EXPERIMENTS AND RESULTS

This section demonstrates the effectiveness of our methodology. First, we have explained the used datasets for both modules including object detection. Next, training methodology along with training setup and implementation details are described. Lastly, training and testing results are explained.

4.1 Dataset

All our modules are pre-trained on the MS COCO dataset. To train for the specialized task of our detection modules, we have used the UA-DETRAC dataset having approximately 83,000 images for training and

validation this dataset, while the test set having around 15,000 images are used to validate the counting performance of both vehicle detectors. It consists of videos captured at 24 different locations at Beijing and Tianjin in China. Both models are trained and validated on the same training and validation set.

4.2 Training Setup

For both the YOLO model, the network is trained for 1,000 epochs, with a batch size of 16, over the UA-DETRAC dataset [34]. To minimize the error, SGD-momentum was used with a learning rate of 0.02, the momentum of 0.8, and weight decay of 0.0002. This Network is trained for 2,000 epochs with batch size 16 For the training of the Faster- RCNN model, Adam optimizer is used having learning rate and weight decay equal to 0.0001. This network is trained for 1, 500 epochs with batch size 16. Xavier initializer was used to initialize the parameters of both models. Both networks were implemented using TensorFlow and Keras on a machine having 16 GB RAM, and an Nvidia 1080Ti GPU. Python 3.7, TensorFlow 1.8, Keras 2.08 and OpenCV 4 are used for the implementation.

4.3 Evaluation

Both models have been evaluated qualitatively, and detection results of the YOLO object detector are also compared with the performance of the Faster-RCNN on the same validation dataset. Loss and accuracy graphs of both models' modules are also given. Fig. 7 is showing some outputs on frame sequence on YOLO and Faster RCNN. Here count is based on the number of vehicles that crossed the marked line.

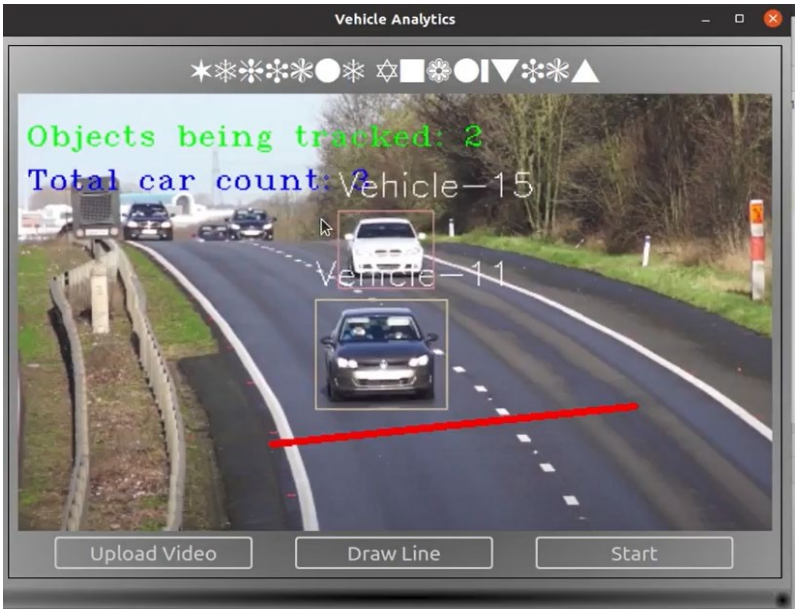

Figure 7.Results on Yolov4

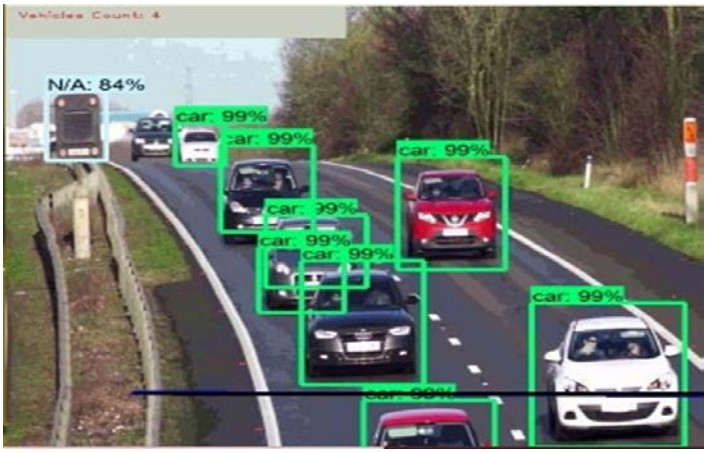

Figure 8. Results on Faster RCNN

We have used pre-trained the YOLOv4 and Faster-RCNN on MS COCO, and the whole network is fine-tuned on the UA-DETRAC dataset having 83k images. Modified- YOLOv4 is trained for 1500 epochs, while the Faster-RCNN model is trained for 2000 epochs and the dataset is divided into training and validation sets with 80%, 20% ratios respectively. From Fig. 9, the Modified-YOLOv4 module also faced overfitting as there is a difference between the training and validation accuracy where validation accuracy remained lower than training accuracy. At the end of the last epoch, training and validation accuracy ended at 95.1% and 89.9% respectively. Training loss (Fig. 9) started reducing from 9.2 approximately, but training loss went to 0.11, and validation loss started from 9.8 ended up at 0.22 at the final epoch. Validation loss went lower to training loss at some epoch but remained high most of the training time.

Figure 9.Training and validation accuracy graph of Modified- YOLOv4

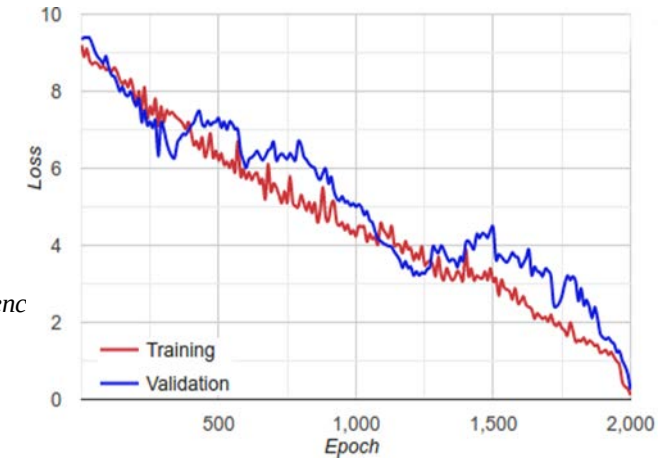

Figure 10.Training and validation loss graph ofModified-YOLOv4

Fig. 10. shows the Faster-RCNN accuracy graph, From the gap of training and validation accuracy, it can be concluded that the model is slightly overfitting the training data which is a curse associated with deep learning models. Training accuracy of Modified-YOLOv4 is 97.2%, while validation accuracy is 94.21%. Faster-RCNN loss for training is started decreasing from 8.6 and lowered to 0.19, while validation loss is reduced from 8.7 to 0.25 after 1500 epochs. A graph in Fig 12 is showing the loss journey of Faster-RCNN throughout training.

Both Networks are overfitted due to different images distribution of dataset for training and validation. With the inclusion of more images of variational image distribution. Moreover, deep learning models are usually very complex due to millions of parameters, so these models have an overfitting effect by nature.

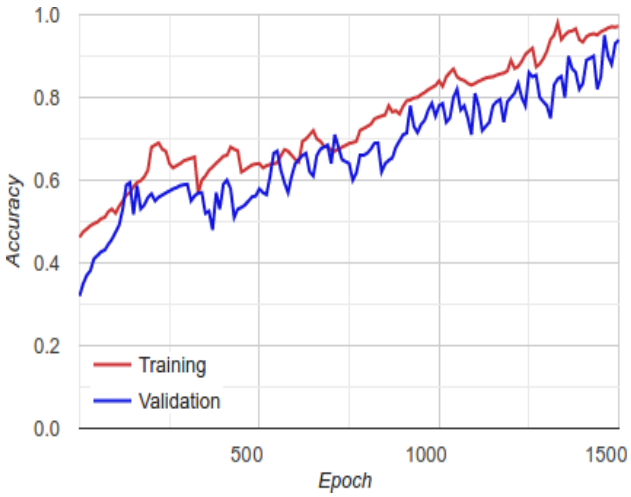

Figure 11.Training and validation Accuracy graph of Faster- RCNN

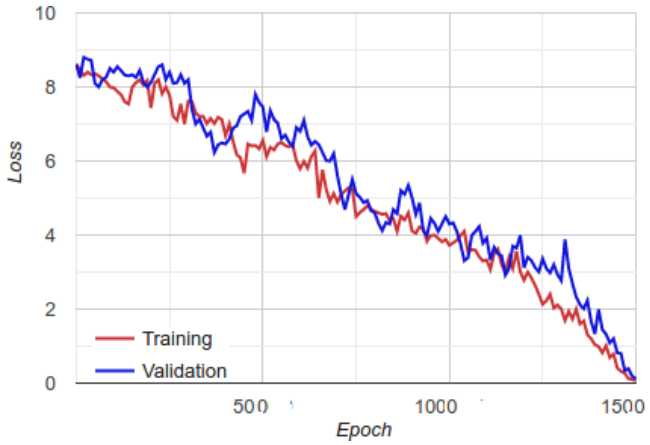

Figure 12.Training and validation Loss graph of Faster-RCNN

Table. 1 gives the breakdown and compares the results of various segment types of the UA-DETRAC dataset. For the SpotNet [35] and both model hard is same having Intel i7 CPU with 32GB RAM and GTX1080 Ti Nvidia GPU. SpotNet [35] has achieved the highest accuracy on the UA- DETRAC dataset. Both models surpass the SpotNet [35] in terms of accuracy among all the dataset segments including hard and cloudy, however, our Faster-RCNN model is very slow as compared to SpotNet. Modified-YOLOv4 is almost 2x faster than SpotNet

Table 1. Accuracy breakdown and comparison on various segments of the dataset.

Model	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny	Speed
SpotNet [35]	86.80%	97.58%	92.57%	76.58%	89.38%	89.53%	80.93%	91.42%	14 fps
Modified- YOLOv4	89.9 %	94.15%	92%	89.14%	91.2%	87.14%	88.21%	87.74%	25 fps
Modified- Faster-RCNN	94.21 %	98.12%	95.14%	93.77%	93.21%	94.12%	92.47%	93.2%	9 fps

5 DISCUSSION

Original YOLOv4-tiny is using cross-stage partial block as a residual block that is compromising the model efficiency while increasing accuracy. To detect the real-time vehicles, a Residual block is used instead of two CSPBlock as in YOLOv4-tiny [1]. This block uses two direct paths network handle input representation map. Compared with CSPBlock used on the original YOLOv4-tiny [29], our ResBlock (Fig.
1) removes the first 3 x 3 Conv in CSPBlock and replaces the consequent 3 x 3 Conv layers with 1 x 1 Conv layers in the Path T network to make the detection network efficient. The proposed ResBlock unit adds pooling and Conv in the Path B network, but this extra computation overhead is very small as compared to reduce in computation in the Path T network. To enhance the CNN performance, two kinds of approaches are used, one is to increase the complexity of the original network by adding more layers, and the other is to change the backbone feature extractor. This approach improves the detection performance more as compared to the first and is more favorable because of the performance of the recent classification network. Inception-Resnet-v2 [30] is a combined form of Inception [31] and Residual network [32]. It replaces the concatenation block in the Inception with the Residual block. This improves the detection performance of the Faster-RCNN due to residual connections without compromising the efficiency of the network.

For the development of this system, work was started on the Google Colab, but the dataset was huge and great

to train, but the free Instance of the Google Colab has the time limitation. Moreover, Google Colab does not offer a user interface to draw the virtual line. Colab was taking so much time to load the dataset that it was very difficult to debug the error, at every error whole dataset was required to reload in the Colab memory.

As the current family of deep learning detection models are designed against fixed resources, so for example Faster- RCNN can give 14 to 17 frames per second. When more and more resources are available, this model will still give 14 to 17 frames per second processing. Moreover, training and testing are required to have only Nvidia GPU enable machine, otherwise, training will take months to complete. Moreover, GPU needs to have memory at least 8 GB to optimally model training. RAM of the machine is also very important because trained is processed in the form of batches of the training dataset. For the batch size, the machine needs to have 16GB RAM.

For the commercial usage of this system, it will be required to have a machine GPU with much more memory and RAM for the training. Moreover, used models will be required to have the ability to scale up on the availability of extra resources. Continual learning will also be required so that system can accommodate vehicles with new shapes. The system will give better performance if the camera would be a better resolution.

6 CONCLUSION

We have used two state-of-the-art object detectors for vehicle counting. YOLOv4 is modified by introducing a Residual Block to make real-time vehicle detection, while Faster- RCNN is modified to vehicle highly accurate vehicle by adding the Inception-ResNet-2v as the backbone. For the Real-time vehicle analytics Modified- YOLOv4is proposed, while for the highly accurate offline analytics Modified Faster-RCNN is proposed. Both detectors are tested on the UA-DETRAC dataset. Modified-YOLOv4 and Faster- RCNN have achieved around 90%, and 95% accuracies respectively. Faster-RCNN is chosen because of its accuracy but as it is very slow, it is used to get the more accurate results offline. On the other hand, YOLOv4 is chosen in the context of production. YOLOv4 is best suitable for real-time processing.

This kind of vehicle analytics can help urban policymakers City Council, City Transport Infrastructure authority to manage traffic efficiently and they can take effective remediation decisions like building footpaths, introductions of one-way roads, construction of new fly overs, expansion of road lanes etc. In the city environment, traffic congestion is a growing issue that leads to wastage of long working hours, consumption of extra fuel that leads to carbon emissions, delay in food and business deliveries, and impacts various other social, environmental, and economic aspects.

The proposed system does not have continual learning, so when a vehicle having a shape that was not part of the training process comes, the system fails to detect and eventually not includes in the counting. The proposed methodology is also designed for the fixed resource regime so when extra resources will be available, this system will be unable to scale up its optimal efficiency.

In future, a model having continual learning and scaled ability concerning resources can be used, it will give real-time processing on any platform independent of the physical hardware. The used UA-DETRAC dataset has certain limitations, it contained specified vehicle categories that operate in the China in a certain traffic conditions , trained model will only be able to detect those kinds of vehicles and with associated traffic conditions. *e.g.* in the dataset don't have much data like traffics in Roundabout, Overpass and Tunnels, our trained model may not perform well in this traffic conditions ,But to make a flexible model that can adopt new vehicles and traffic conditions without retraining the model from scratch, continual learning will give this ability.

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, ‘‘Datadriven intelligent transportation systems: A survey,’’ IEEE Trans. Intell. Transp. Syst., vol. 12, no. 4, pp. 1624–1639, Dec. 2011

[2] A. M. Pereira, ‘‘Traffic signal control for connected and non- connected vehicles,’’ in Proc. Smart City Symp. Prague (SCSP), Prague, Czech Republic, May 2018, pp. 1–6.

[3] Dai Z, Song H, Wang X, Fang Y, Yun X, Zhang Z, Li H. Video- based vehicle counting framework. IEEE Access. 2019 May 1;7:64460-70.

[4] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) 2005 Jun 20 (Vol. 1, pp. 886-893). Ieee

[5] Mita T, Kaneko T, Hori O. Joint haar-like features for face detection. InTenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 2005 Oct 17 (Vol. 2, pp. 1619-1626). IEEE.

[6] L. Juan and O. Gwon, A comparison of SIFT, PCA-SIFT and SURF,’’ Image Process. Pattern Recognit., vol. 3, no. 4, pp. 143–152, 2009

[7] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 580-587).

[8] Girshick R. Fast r-cnn. InProceedings of the IEEE international conference on computer vision 2015 (pp. 1440-1448).

[9] Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real- time object detection with region proposal networks. InAdvances in neural information processing systems 2015 (pp. 91-99).

[10] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 779-788).

[11] Redmon J, Farhadi A. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767. 2018 Apr 8.

[12] Bochkovskiy A, Wang CY, Liao HY. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. 2020 Apr 23.

[13] Jiao L, Zhang F, Liu F, Yang S, Li L, Feng Z, Qu R. A survey of deep learning-based object detection. IEEE Access. 2019 Sep 5;7:128837-68.

[14] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015 May;521(7553):436-44.

[15] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

[16] Oquab M, Bottou L, Laptev I, Sivic J. Weakly supervised object recognition with convolutional neural networks. InProc. of NIPS 2014 Jun (pp. 1545-5963).

[17] Oquab M, Bottou L, Laptev I, Sivic J. Learning and transferring mid-level image representations using convolutional neural networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 1717-1724)

[18] Finney DJ. Probit analysis: a statistical treatment of the sigmoid response curve. Cambridge university press, Cambridge; 1952.

[19] Kavukcuoglu K, Ranzato MA, Fergus R, LeCun Y. Learning invariant features through topographic filter maps. In2009 IEEE Conference on Computer Vision and Pattern Recognition 2009 Jun 20 (pp. 1605-1612). IEEE.

[20] He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence. 2015 Jan 9;37(9):1904-16.

[21] Dai J, Li Y, He K, Sun J. R-fcn: Object detection via region- based fully convolutional networks. InAdvances in neural information processing systems 2016 (pp. 379-387).

[22] Yoo D, Park S, Lee JY, Paek AS, So Kweon I. Attentionnet: Aggregating weak directions for accurate object detection. InProceedings of the IEEE International Conference on Computer Vision 2015 (pp. 2659-2667).

[23] Erhan D, Szegedy C, Toshev A, Anguelov D. Scalable object detection using deep neural networks. InProceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 2147-2154).

[24] Najibi M, Rastegari M, Davis LS. G-cnn: an iterative grid based object detector. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 2369- 2377).

[25] Huval B, Wang T, Tandon S, Kiske J, Song W, Pazhayampallil J, Andriluka M, Rajpurkar P, Migimatsu T, Cheng-Yue R, Mujica F. An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716. 2015 Apr 7.

[26] Choi J, Chun D, Kim H, Lee HJ. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. InProceedings of the IEEE International Conference on Computer Vision 2019 (pp. 502-511).

[27] Hu HN, Cai QZ, Wang D, Lin J, Sun M, Krahenbuhl P, Darrell T, Yu F. Joint monocular 3D vehicle detection and tracking. InProceedings of the IEEE international conference on computer vision 2019 (pp. 5390-5399).

[28] Cao J, Song C, Song S, Peng S, Wang D, Shao Y, Xiao F. Front Vehicle Detection Algorithm for Smart Car Based on Improved SSD Model. Sensors. 2020 Jan;20(16):4646.

[29] Alexey Bochkovskiy. Darknet: Open Source Neural Networks in Python. 2021. Available online: https://github.com/AlexeyAB/darknet (accessed on 26 May 2021).

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings o f the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.

[31] Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning. InProceedings of the AAAI Conference on Artificial Intelligence 2017 Feb 12 (Vol. 31, No. 1).

[32] Targ S, Almeida D, Lyman K. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029. 2016 Mar 25

[33] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft coco: Common objects in context. InEuropean conference on computer vision 2014 Sep 6 (pp. 740-755). Springer, Cham.

[34] Wen L, Du D, Cai Z, Lei Z, Chang MC, Qi H, Lim J, Yang MH, Lyu S. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding. 2020 Apr 1;193:102907.

[35] Perreault H, Bilodeau GA, Saunier N, Héritier M. Spotnet: Self-attention multi-task network for object detection. In2020 17th Conference on Computer and Robot Vision (CRV) 2020 May 13 (pp. 230-237). IEEE.

Jiang Z, Zhao L, Li S, Jia Y. Real-time object detection method based on improved YOLOv4-tiny. arXiv preprint arXiv:2011.04244. 2020 Nov 9.