Efficient Vehicle Status Network with Adaptive Adversarial Learning

Youlkyeong Lee, Jehwan Choi, Kanghyun Jo
Dept. of Electrical, Electronic and Computer Engineering
University of Ulsan, Ulsan, Republic of Korea
yklee00815@gmail.com, {jehwan, acejo}@ulsan.ac.kr

***Abstract*—The rapidly developing autonomous driving field now needs a more secure transportation system through information between multiple mobility. Deep learning that can judge traffic conditions by convergence of various sensor data and in particular, research on the convolutional neural network using computer vision are being actively conducted. In addition, recognizing many objects at once in a large area through drone images and understanding the movement of the object is used as safe traffic assistance information. In this study, an image classification study is conducted to determine the status of the vehicle on the road through drone flight image data. The goal is to build a new image classification model robust to the proposed image classification network by applying the weighted adversarial learning method. Weight adversarial learning is a method of securing robust performance in image classification of various statuses while disturbing the model by forcibly reflecting the slope value in reverse when updating the network through the reverse gradient layer. In the experiment, model performance is evaluated through the collected drone flight data set.**

***Index Terms*—Adaptive adversarial learning, classification, drone image, transportation system, vehicle status**

I. Introduction

1) Autonomous Driving Mobility: Autonomous driving systems are developing at a remarkable pace, and shortly, we can expect to utilize safe autonomous vehicle services. This will likely be achieved through the fusion of various data types, such as traffic signals, road signs, and vehicle information. Moreover, the movement data of vehicles on the road (buses, cars, motorcycles, trucks, etc.) and data collected by fixed traffic information devices (Closed-circuit Television, CCTV) will be complemented through data sharing with future mobility technologies like drones. This approach will extend road information collection and analysis. Especially, flying drones, equipped with cameras and LiDAR sensors, have the advantage of capturing wide areas from a high altitude in a single shot, acquiring dynamic environmental information on the roads. This overcomes the limited data collection capabilities of on-road vehicles and provides effective information to autonomous driving assistance systems, playing a crucial hardware role in establishing the next generation of road safety networks.

2) Convolutional Neural Networks: The recent surge in research on models developed based on artificial neural networks, particularly Convolutional Neural Networks (CNNs), highlights their popularity in image-based artificial intelligence model development. The diversity of real-world data requires

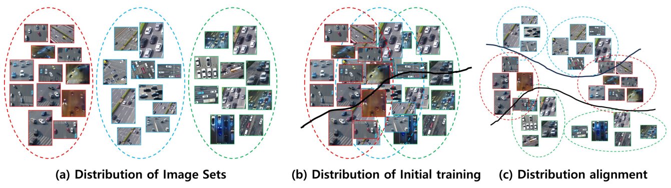

(a) Distribution of Image Sets
(b) Distribution of Initial training
(c) Distribution alignment

Fig. 1: Vehicle Status Data Distribution

the computation of deep convolutional layers, mirroring the human neural network structure. Furthermore, widely used for tasks like image classification and lightweight model design, the MobileNet series [1], [2], SqueezeNet [3], ShuffleNet [4], among others, offer high performance and are frequently deployed on mobile devices. This allows these models to perform critical functions such as situation assessment, recognition, and prediction. Consequently, CNNs have become an essential module in building autonomous driving assistance systems, garnering intense interest and becoming a hot topic among many researchers.

3) Proposed Adversarial Learning: In this research, as an alternative method to improve performance, a weight adversarial training method is proposed. Fig. 1 shows how confusing each class of dataset. The picture illustrates simple road conditions, texture, types of vehicles, weather, etc. This approach robustly overcomes the diversity of real-world road image data. It involves adding noise to the feature maps extracted within the network, thereby incorporating feature diversity into the network's weights and updating the learning model. This method allows the model to adapt to new image information without the need to continually add new image data for training. The experimentation involves using a drone flight dataset, which captures road images, to evaluate this training method on a model designed to determine vehicle status.

This weight adversarial training approach offers several advantages. Firstly, it reduces the reliance on extensive data augmentation techniques, saving time and computational resources. Secondly, embedding diversity directly into the model's weights, enhances the model's generalization capabilities, making it more effective in handling unseen or novel scenarios. This is particularly relevant for autonomous driving systems, where the ability to adapt to a wide range of driving conditions and environments is crucial. Overall, this proposed

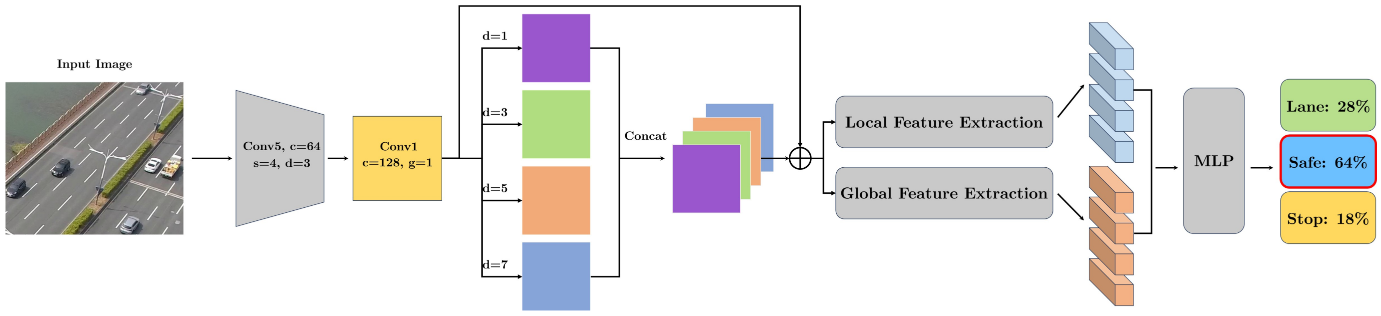


Fig. 2: Overall architecture for efficient vehicle status classification.

method represents a significant step forward in developing more efficient and robust models for real-world applications. The contributions of the proposed methods are as follows:

- Wide Area Feature Extraction: Four types of dilated convolutional layers are combined to create the initial feature extraction module
- Local and Global Feature Extraction: Deformable convolutional layers are used for regional feature extraction, while self-attention mechanisms are employed for global semantic feature extraction
- Adaptive Adversarial Learning: Adding in-layer random noise for weighted adversarial learning contributes to a more robust distribution of results with four dilated convolution layers

II. Proposed Methods

A. Overall Architecture

The proposed vehicle status classification algorithm adopts the approach of Lee et al. [5] in Fig. 2. As suggested in VSNet, for vehicle detection, the YOLOv5 model [6] is used on drone flight datasets [7], [8]. The detected vehicles are then cropped along with their surrounding areas to be used as input images. The vehicle status is classified into three categories: Normal, Stopped, and Lane Changing. The proposed algorithm suggests a model that can adapt to the dataset autonomously, proposing an adaptive adversarial learning model for determining vehicle status applied with four dilated convolution layers.

B. Wide Area Feature Extraction

The recent feature extraction in various classification models has involved layers that are deep and combine layers that extract features with various kernel sizes, such as VGG [9], ResNet [10], Inception [11]. Efforts have been made to generate good feature maps in the initial stages of the model structure. These models are trained on the ImageNet dataset [12], which is a general-purpose dataset for common objects, thus possessing high versatility. However, the purpose of feature extraction varies depending on the model's characteristics. In this research, feature extraction is conducted for image data with wide areas and small objects distanced from each other, like drone flight footage. This is referred to as the Wide Area Feature Extraction (WAFE) module. Table. 1 describes the layers that make up the WAFE module. The initial block of the

TABLE 1: Setting of WAFE module. **c**: channel, **k**: kernel size, **s**: stride, **d**: dilated ratio, **g**: group, **p**: padding

Layer Name	**Output**	**c**	**k**	**s**	**d**	**g**	**p**
Input	512×512	3	-	-	-	-	-
Conv5	128×128	64	5	4	3	4	6
Conv1	128×128	128	1	1	1	4	-
BN	128×128	128	-	-	-	-	-
GELU	128×128	128	-	-	-	-	-
Dropout	128×128	128	-	-	-	-	-
DConv3_1,BN,GELU	128×128	32	3	1	1	1	1
DConv3_2,BN,GELU	128×128	32	3	1	3	1	3
DConv3_3,BN,GELU	128×128	32	3	1	5	1	5
DConv3_4,BN,GELU	128×128	32	3	1	7	1	7
Concatenate	128×128	128	-	-	-	-	-
Conv1	128×128	128	1	1	1	4	-
Addition	128×128	128	-	-	-	-	-

WAFE module consists of two convolutional layers. The first layer has a kernel size of 5×5, channels=64, a dilated ratio=4, padding=6, and a stride of 4. The second layer employs a 1×1 convolutional layer and batch normalization, adopting the GELU [13] as the activation function. The characteristics of drone flight data include vehicles in the image being widely spaced. To overcome this in feature extraction, the dilated convolution filter proposed by Liang-Chieh et al. [14] is applied. The dilated ratio expands the kernel size, allowing reference to a wide receptive field of pixel information while maintaining efficient computation, contributing to effective feature extraction. In this module, four types of dilated ratios, $d = [1, 3, 5, 7]$, are applied, dividing the input value into four equal parts. These four methods form various feature maps, constituting an important part of feature extraction in the proposed architecture.

C. Object-oriented Feature Extraction

In this part of the study, the focus is on extracting more robust features from the feature maps derived from the backbone, which contains the overall information of the image. Robust features are likely to be located especially around the characteristics of objects in the image. When conventional convolutional layers are applied, the features are computed at fixed receptive field locations as calculated in Eq. (1).

$$Cov(p_0) = \sum_{p_n \in \mathcal{R}} W(p_n) \cdot I(p_0 + p_n) \tag{1}$$

In the feature map, p_n refers to the position where the calculation with the kernel occurs. For a 3×3 kernel, this range is represented as $\mathcal{R} = (-1,1), (-1,0), (0,0), \ldots, (1,0), (1,1)$. p_0 denotes the initial position of the kernel, which is represented in (x, y) coordinates. Stacking multiple layers in this manner results in the observation of unnecessary features at all locations, including the background, multiple times. To compute this more effectively, the deformable convolutional layer [15] is applied. The deformable convolution is expressed in Eq. (2) as follows.

$$DConv(p_0) = \sum_{p_n \in \mathcal{R}} W(p_n) \cdot I(p_0 + p_n + \Delta p_n) \tag{2}$$

In the existing kernel position, Δp_n is added to modify the location where the feature map is computed. This value is applied as a learnable parameter, allowing the position to update around robust features. Such an approach enhances the weights of the layer that extracts features based on important feature values within the feature map. For extracting features centered around objects in the feature map, Object-oriented Feature Extraction is depicted as follows. Initially, two Deformable Convolutional (DConv) layers are stacked. Then, a 3×3 and a 1×1 convolution are applied to share feature information. Next, the feature map is halved through Maxpooling, and two additional DConv layers are applied to form an object-centered feature extraction module. This module plays a role in enhancing the understanding of the image based on the robust features possessed by objects in the feature map.

D. Global-oriented Feature Extraction

The repeated use of convolutional layers often leads to the final feature maps of deep layers focusing too narrowly on specific parts of the original image. This localized focus can result in inaccuracies in image classification decisions. To address this issue, an approach that differs from Object-oriented feature extraction, which primarily focuses on local features, is employed. This approach involves using the encoder part of the Vision Transformer (ViT) [16]. The key concept of this method is to extract features by utilizing the entire input image to assess similarities among global pixel information. The Transformer encoder utilizes the feature maps extracted from the backbone network as its input. To handle 2D images, these extracted feature maps, represented as $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, are rearranged considering the image size (H, W), the channel size C, and the patch size (P, P). Here, $N = (H \times W)/P^2$ represents the number of patches, and the feature map is rearranged into $\mathbf{x_p} \in \mathbb{R}^{N \times (P^2 \cdot C)}$. This is made into the efficient sequential length of the Transformer, providing a trainable linear projection in the D dimension. Next, the missing position information from the value converted into a vector is included by adding positional encoding (Eq. (3), z_0). The encoder is composed of multi-head self-attention (MSA) and multi-layer perception (MLP) blocks (Eq. (4), and Eq. (5)).

$$\begin{gathered}\mathbf{z}_0 = [\mathbf{x_p^1 E}; \mathbf{x_p^2 E}; \mathbf{x_p^3 E}; , \cdots, \mathbf{x_p^N E}] + \mathbf{E}_{pos}, \\ \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}\end{gathered} \tag{3}$$

$$\mathbf{z}_l^{'} = \mathbf{MSA}(\mathbf{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, l = 1, \ldots, L \tag{4}$$

$$\mathbf{z}_l = \mathbf{MLP}(\mathbf{LN}(\mathbf{z}_l^{'})) + \mathbf{z}_l^{'}, l = 1, \ldots, L \tag{5}$$

Ultimately, the Global-oriented Feature Extraction divides the entire feature map into a grid, computing global features by analyzing the correlations between grids, and providing these for image classification decisions.

E. Adaptive Adversarial Learning

In Neural Networks (NN), the number of parameters that need to be handled is immense. It is impossible for humans to manually adjust these parameters to optimize the model. Instead of directly modifying the network's weights, adversarial training can be constructed by manipulating the input images or feature maps. The Reverse Gradient Layer [17], denoted as $R_\lambda(\mathbf{x})$, consists of forward and back propagation (Eq. (6) and Eq. (7)).

$$R_\lambda(\mathbf{x}) = \mathbf{x} \tag{6}$$

$$\frac{dR_\lambda}{d\mathbf{x}} = -\lambda \mathbf{I} \tag{7}$$

The proposed method is to forcibly add noise to the weight of the existing layer to find robust weights in various image environments in the future.

$$\tilde{A}^l = (1 - \alpha)N + \alpha A^l \tag{8}$$

In Eq. 8, A^l is a feature map for the l-th layer, and N is a noise map of the same size as A^l. α generates $\tilde{A}^l$ with the feature values of the feature map and the noise map as weights. α is a value between 0 and 1. And when one 3×3 Conv layer is applied and backpropagation proceeds, a reverse gradient layer is applied to update the Eq. (9) weight in a different direction for the amount of change for that layer.

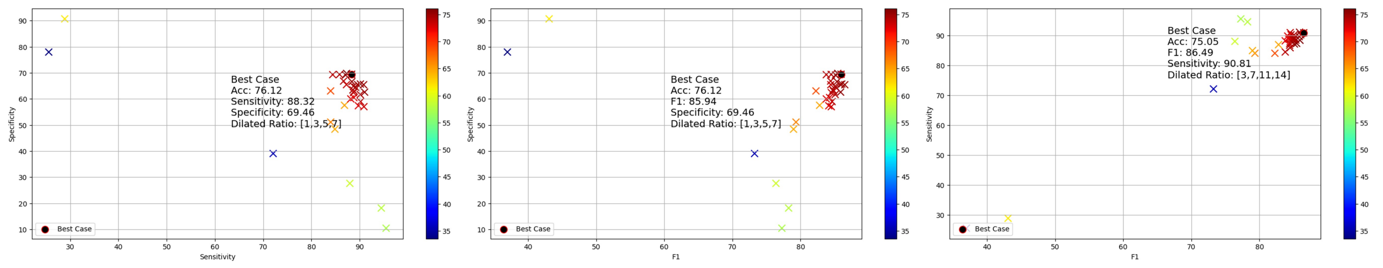
$$\bar{A}^l = \tilde{A}^l + \beta \frac{\partial \mathcal{L}_{total}}{\partial \tilde{A}^l} \tag{9}$$

$\bar{A}^l$ is a feature map updated after applying the reverse gradient layer. β proposes adaptive adaptive adaptive learning by applying a weight of gradient change at a rate of 0.99 per epoch.

III. Experiment

A. Implementation Details

The vehicle status detection model is developed based on the PyTorch [18] library. The input image is size-adjusted to 512×512. The Batch size is set to 64. The optimization function uses SGD [19] and the learning rate is 0.01. α generates a random value within the beta distribution. β reduces the ratio by 0.99 for each epoch. The GPU uses 4 Nvidia A100 sheets.



(a) Specificity-Sensitivity Distribution (b) Specificity-F1 Distribution (c) Sensitivity-F1 Distribution

Fig. 3: To figure out the best model with several metrics, the graphs show the various combinations of conditions for vehicle status classification performance.

B. Datasets

There are few public datasets for vehicle status classification using drones. In addition, since various drone datasets do not serve the purpose of vehicle status classification, datasets must be collected and processed directly. There are three types of vehicle status classification: stop, lane change, and normal. This dataset uses the drone flight dataset used in [7], [8]. As input images of the vehicle classification model, some of the images are cut out to build a dataset through the method proposed by Lee et al. [5], [7], [8] for the target vehicle. The total number of images is 14,906, and the training and test images are divided into 8:2.

C. Metrics

There are main metrics in this study: Accuracy, specificity, sensitivity, and F1 score.

- **Accuracy**
  - This metric indicates how well the model classifies. It is the proportion of correctly classified data out of the total data.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

- **Specificity**
  - This metric indicates how accurately the model classifies negative instances among the actual negative cases.

$$\textbf{Specificity} = \frac{TN}{TP + FP} \tag{11}$$

- **Sensitivity**
  - This metric indicates how accurately the model classifies positive instances among the actual positive cases.

$$\textbf{Sensitivity} = \frac{TP}{TP + FN} \tag{12}$$

- **F1 Score**
  - This is the harmonic mean of Precision and Recall, and it considers the balance between these two metrics.

$$\textbf{F1 score} = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{13}$$

D. Comparison Results

The performance of proposed vehicle status classification models is compared using widely used image classification models [1], [2], [3], [4], [20], [21], as shown in Table 2. $\mathbf{w}$ denotes pre-trained models trained on ImageNet [12], while $\mathbf{w/o}$ represents those without pre-training. Generally, pre-trained models on a large dataset like ImageNet tend to outperform the proposed models. The proposed models utilize the encoder of a transformer within them, requiring parameters such as multi-head and embedding size within the self-attention mechanism. Among the existing $\mathbf{w/o}$ models, MobileNetV3 [2] exhibits the highest accuracy at 79.54%, showing a performance difference of 3.42% compared to the proposed model ($\mathbf{h} = 32, \mathbf{e} = 128$). Although the proposed models show lower performance compared to the MobileNet series [1], [2], they demonstrate higher accuracy than the other models. Semantic evaluation of the models is conducted using evaluation metrics such as Sensitivity, Specificity, and F1 score, as illustrated in Fig. 3. This involves plotting performance metrics against various combinations of hyperparameters to visualize their distributions. Models located in the upper-right corner of this graph are considered to have stable performance. Under the dilated ratio conditions of 1, 3, 5, and 7, the proposed model with ($\mathbf{h} = 32, \mathbf{e} = 128$) is situated at the upper-right corner in the Specificity-Sensitivity and Specificity-F1 distributions, but not in the Sensitivity-F1 distribution. This highlights that elevated accuracy does not automatically ensure consistent model performance.

E. Runtime Results

The aim is to construct an efficient model for deployment on mobile devices in the future. Table 3 presents the results of runtime measurements. Most proposed models achieve around 500 FPS. Efficient speed is achieved through parallel computation via group convolution layers within the model design. For instance, in SqueezeNet, 128 channels are reduced to 16 channels through a 1×1 conv layer, followed by two branches producing 128-channel output: one with 64 1×1 conv filters and the other with 64 3×3 conv filters. This drastically reduces parameters but leads to higher GFlops compared to the proposed model, resulting in lower accuracy. Given the importance of both fast decision-making and high accuracy in determining traffic conditions, a balanced research approach is required between these two metrics.

IV. Conclusion

This study proposes a vehicle status classification network based on weighted adversarial learning using drone flight image data. The proposed vehicle status classification model consists of wide-area feature extraction, object-oriented feature extraction, and global-oriented feature extraction. The proposed weight adversarial learning is applied to each module. The forcibly added noise map allows learning to proceed by adjusting the gradient reflection ratio as the echo proceeds. It is currently experimenting and producing meaningful results.

Acknowledgment

This result was supported by ”Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003)

References

[1] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[2] Andrew G. Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[3] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. *ArXiv*, abs/1602.07360, 2016.

[4] Ningning Ma, Xiangyu Zhang, Haitao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *ArXiv*, abs/1807.11164, 2018.

[5] Youlkyeong Lee, Jehwan Choi, and Kanghyun Jo. Vsnet: Vehicle state classification for drone image with mosaic augmentation and soft-label assignment. In *Asian Conference on Intelligent Information and Database Systems*, 2023.

[6] Glenn R. Jocher et al. ultralytics/yolov5: v5.0 - yolov5-p6 1280 models, aws, supervise.ly and youtube integrations. 2021.

[7] Youlkyeong Lee, Seong Eun Kim, Jehwan Choi, and Kanghyun Jo. Vehicle state classification from drone image. *2023 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–5, 2023.

[8] Youlkyeong Lee, Jehwan Choi, Jinsu An, and Kang-Hyun Jo. Csa: Channel-wise similarity attention for vehicle state classification. *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, pages 01–05, 2023.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2015.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211 – 252, 2014.

[13] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin P. Murphy, and Alan Loddon Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2016.

[15] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

[17] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, 2014.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[19] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[20] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019.

[21] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2016.

TABLE 2: Comparison of Vehicle Status Classification Model. There are two types of a state-of-the-art classification model(The first group is a light model, the second is a heavy model.

Method	**Accuracy(%)**		**Sensitivity(%)**		**Specificity(%)**		**F1 score(%)**		**Precision(%)**	
	w	**w/o**	**w**	**w/o**	**w**	**w/o**	**w**	**w/o**	**w**	**w/o**
MobileNetv2 [1]	83.10	78.50	90.00	86.10	84.17	79.47	90.46	86.96	90.93	87.83
MobileNetv3_small [2]	79.68	77.63	89.74	91.65	74.85	67.24	87.89	87.00	86.10	82.80
MobileNetv3_large [2]	80.28	79.54	92.62	94.57	69.78	65.55	88.28	88.37	84.33	82.93
SqueezeNet [3]	81.96	71.36	89.41	86.01	81.82	60.07	89.50	82.36	89.59	79.02
ShuffleNetv2 [4]	82.53	71.97	94.76	88.09	74.25	71.79	90.46	85.45	86.53	82.97
Efficient_b0 [20]	82.56	73.71	90.46	88.15	84.26	63.80	90.68	84.20	90.90	80.60
Efficient_b4 [20]	83.10	67.98	91.34	96.15	83.07	45.07	90.83	83.73	90.32	74.15
Efficient_wideses_b0 [20]	82.96	73.84	90.98	88.75	82.67	61.78	90.54	84.25	90.10	80.17
Efficient_wideses_b4 [20]	84.17	72.60	91.76	90.10	83.11	59.04	91.13	84.01	90.52	78.70
ResNeXt-50 [21]	84.04	78.64	92.00	92.49	81.03	66.63	90.77	87.55	89.57	83.11
ResNeXt-101 [21]	84.57	78.07	92.47	87.09	82.95	75.15	91.51	86.53	90.57	95.97
Proposed(h=8, e=96)	-	76.09	-	87.28	-	69.76	-	56.46	-	83.71
Proposed(h=16,e=96)	-	74.21	-	90.10	-	61.68	-	85.08	-	80.59
Proposed(h=32,e=96)	-	74.04	-	86.65	-	66.83	-	84.42	-	82.30
Proposed(h=8, e=128)	-	75.35	-	87.07	-	69.15	-	85.13	-	83.28
Proposed(h=16,e=128)	-	73.31	-	87.25	-	65.45	-	84.44	-	81.81
Proposed(h=32,e=128)	-	76.12	-	88.32	-	69.46	-	85.94	-	83.69

TABLE 3: Comparison of runtime result with the state-of-the-art.

Method	**FPS**	**GFlops**	**Params(M)**	**Head**	**Emb**	**Patch**
MobileNetv2 [1]	435	1.67	2.23	-	-	-
MobileNetv3_small [2]	471	0.30	1.52	-	-	-
MobileNetv3_large [2]	370	1.17	4.21	-	-	-
SqueezeNet [3]	1201	3.99	0.74	-	-	-
ShuffleNetv2 [4]	422	0.77	1.26	-	-	-
Efficient_b0 [20]	347	2.08	4.01	-	-	-
Efficient_b4 [20]	192	8.01	17.55	-	-	-
Efficient_wideses_b0 [20]	310	2.08	7.15	-	-	-
Efficient_wideses_b4 [20]	178	8.03	33.14	-	-	-
ResNeXt-50 [21]	539	22.24	22.99	-	-	-
ResNeXt-101 [21]	306	86.07	86.74	-	-	-
Proposed(h=8, e=96)	681	1.36	3.06	8	96	16
Proposed(h=16,e=96)	522	1.36	3.06	16	96	16
Proposed(h=32,e=96)	510	1.36	3.06	32	96	16
Proposed(h=8, e=128)	450	1.73	3.53	8	128	16
Proposed(h=16,e=128)	624	1.73	3.53	16	128	16
Proposed(h=32,e=128)	576	1.73	3.53	32	128	16