Received 7 April 2025, accepted 8 May 2025, date of publication 27 May 2025, date of current version 10 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3574187

RESEARCH ARTICLE

Minimizing Model Size of CNN-Based Vehicle Make Recognition for Frontal Vehicle Images

WIPUT PUISAMLEE[1] **AND RATHACHAI CHAWUTHAI**[2]
[1]Department of Electrical Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
[2]Department of Computer Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
Corresponding author: Rathachai Chawuthai (rathachai.ch@kmitl.ac.th)

ABSTRACT Vehicle Make Model Recognition (VMMR) is commonly used in Intelligent Transportation Systems (ITS), free-flow image-based toll systems, and enforcement systems. These systems must analyze and process vehicle front images for use as evidence. Convolutional Neural Networks (CNN) are widely used for image classification and VMMR problems. Complex model structures and more internal parameters are needed to improve classification accuracy with many classes. Issues included larger models and longer processing times. The goal of this work is to study and create a smaller CNN model that can be used on devices with limited resources, like embedded computers and embedded computer cameras, to figure out what kind of car it is from a front view picture. Real free-flow toll systems were used to train a CNN model that recognized vehicle makes with 99% accuracy. The model is smaller than VGG16, InceptionV3, Yolo11m-cls, and ResNet50 and has over 90% accuracy. It reduced parameters by 69.95% and developed the CTv1 model to achieve an F1 score 2.06% higher than InceptionV3, the best. The model was tested on a Raspberry Pi 3 Model B, processing images in 1 second and using 25 mWh. The compact version of the proposed model also adjusts the Padding and Stride of the Convolutional Layer and reduces the CNN model size using Depth-wise Separable Convolutional and 1×1 Convolutional Dimension Reduction (Bottleneck) methods to test vehicle make recognition accuracy, training time, processing time, and model size.

INDEX TERMS Deep learning, convolutional neural network, intelligent transportation system, vehicle make, model recognition.

I. INTRODUCTION

Current Vehicle Make Model Recognition (VMMR) technologies remain a significant area of interest for researchers and industries, since they play a crucial role in traffic management, tolling, and vehicle re-identification. Specifically, the free-flow tolling system worked with license plate recognition from frontal vehicle images. The current system in Thailand uses only the license plate to identify an individual car and generate an invoice, and it also uses the detected vehicle make and model to help the operator confirm the vehicle information when the system cannot identify the license. Moreover, the free-flow expressway system that is used in Thailand has a relatively high installation cost because detecting and classifying vehicles requires high-performance processing hardware. However, relying on only license plate recognition, the system may face the problems of misreading unclear license plates and vehicle registration fraud. Therefore, using other attributes of the vehicle to consider, such as the brand of the vehicle, may help charge the service fee for the correct vehicle. In addition, the result can lead to the arrest of fraud related to illegal vehicle registrations. Over the past decade, machine learning techniques have garnered substantial attention and development from any research studies, and they were applied across various fields. Specifically, Convolutional Neural Networks (CNNs) are developed for image classification and recognition, and it is also applied to VMMR to enhance performance [1], [2], [3], [4], [5], [6]. Some studies have incorporated state-of-the-art models like ImageNet, VCC16, Inception, Yolo (you only look once), and ResNet [7], [8], [9], [10], [11], [12], [13], [14], [15] to improve classification accuracy. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Antonio J. R. Neves.

achieving high performance is often due to the larger model sizes and higher computational costs, which require more computing resources [16] leading to a high-cost solution. Most pieces of research utilize standard VMMR datasets, and they typically consist of images of entire vehicle. However, due to the real-world scenario of toll plazas in Thailand, image data in this study consisted of long-range photographs of vehicle fronts at toll plazas including their license plate (FIGURE 6). Using our dataset for testing with state-of-the-art models showed varying accuracy values, with some models achieving over 90% accuracy and others below 50%. Experiments showed larger models slowed processing on the Raspberry Pi 3 Model B.

To address an according issue, this study focuses on the classification solution. This study presents a CNN model for recognizing and classifying vehicle makes from front-view images, designed to be small enough for resource-constrained systems. The study involved creating a small CNN model, experimenting with different layer configurations, and adjusting key hyperparameters related to CNNs. The resulting filter-block structure achieved over 95% accuracy in vehicle make classification and was smaller than state-of-the-art models. With the small CNN model size, the demand for processing resources is reduced, and the costs associated with these resources are also lowered.

The research report is divided into four main sections: Related Work, which discusses various studies on VMMR technology that have improved CNNs; Proposed Model, which introduces the CNN model for VMMR, detailing the dataset used for training and testing; Performance Evaluation, which compares the proposed model's performance in vehicle make classification against state-of-the-art models (ImageNet, VCC16, Inception, ResNet, MobileNet), including model size and training time; and Conclusion.

II. LITERATURE REVIEW

A. RELATE VMMR CNN-BASE WORK

There's significant research on applying machine learning, particularly CNN technology, for classifying vehicle make and model [4], [17]. Over the past decade, many studies have experimented with well-known state-of-the-art CNN models [18], [19], [20], [21] and even enhanced methods beyond CNN [22]. Common datasets for these experiments include Stanford Cars, CompCars, and CompCars Surveillance [23], offering accessible resources for evaluating classification results and efficiency.

VGG-oriented The VGG16 model was employed by Naseer et al. [7] as a feature extractor. They then used data from the FC-6 layer to test SVM, Tree, and KNNs for classification. SVM classification produced higher accuracy than direct VGG16 classification, according to their tests on the NTOU-MMR dataset. Among other CNN models, Liu and Wang [8] investigated VGGNet for vehicle make and model classification. They looked into different weight configurations, such as training new weights from scratch or using the original weights completely or in part.

Methods based on Inception, The InceptionV3 model has been used in numerous studies. In order to classify vehicle make and model from nighttime near-infrared (NIR) images, Balci et al. [9] experimented with InceptionV3. They investigated preprocessing techniques like LIME, Ying, and LightenNet for enhancing low-light images and preprocessed normal images using gray or dark filters to mimic NIR images prior to training. Their results demonstrated that the best accuracy was obtained when Ying's preprocessing was applied to dark-filtered images. Re-training make and model classes within the InceptionV3 model when adding new classes with few sample images was the subject of another study by Balci and Artan [10]. They used the Nearest Class Mean Classifier and Weight Imprinting Based Classifier techniques to assess the model's feature embedding outcomes. Using their developed Lightweight Recurrent Attention Unit (LRAU), Boukerche and Ma [11] also tested InceptionV3 (along with ResNet50 and VGG16) to improve classification performance on well-known VMMR datasets such as Stanford Cars, CompCars, and NTOU-MMR.

ResNet-based Approaches: To enhance vehicle make and model classification across multiple VMMR datasets, Boukerche and Ma [11] applied their LRAU to the ResNet50 model (along with VGG16 and InceptionV3). Their findings demonstrated that LRAU could improve classification accuracy. In order to lower the classification accuracy of CNN-based VMMR models, Siddiqui and Ma [12] trained the ResNet50 model to generate simulated occluded portions of vehicle images. In response, they suggested a Symmetric Image-Half Flip preprocessing technique to deal with the issue of occlusion.

YOLO-oriented Methods: Shuo et al. [24] used Yolo for Vehicle Logo Detection (VLD), which focuses on vehicle classification. They developed a customized dataset with car logos from 30 brands and altered Yolo to enhance its small object detection capabilities. In comparison to other models, their tests showed that the modified Yolo model produced the best accuracy and the fastest processing time.

These studies demonstrate the broad use and ongoing advancement of CNN technology for the classification and recognition of vehicle makes and models.

B. CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (CNN) is a type of artificial intelligence model [13] that is well-suited for processing or recognizing image data. The CNN model is developed from the foundation of the ANN model, inspired by the way the human visual system works to see and recognize various images. It consists of two parts: the first part involves seeing and distinguishing key features in the image, and the second part involves learning and remembering these key features for interpretation. A simple CNN model consists of the following picture below.

From the basic structure of a CNN model, numerous studies have developed various structures to suit specific

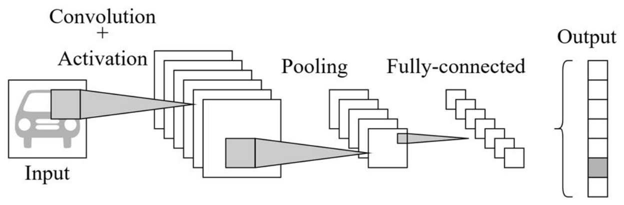


FIGURE 1. A general CNN structure consists of a convolution layer for extracting feature mapping with activators. The next step is a pooling layer, which selects the most significant feature to form the fully connected layer. Finally, the output of prediction.

objectives and problems. State-of-the-art CNN models also have their foundations in simple CNN models.

C. DEPTH-WISE SEPARABLE CONVOLUTION

The structure of some CNN models is designed to recognize a large number of classes, such as 1000 classes (VGG16, ResNet50, Inception). Recognizing and distinguishing a large number of classes requires an internal structure with a sufficient number of layers and filter outputs (results from convolutional operations) to differentiate the various features of many classes. This results in large models that require high computational resources. Depth-wise Separable Convolution [14] is a method that separates the convolutional operation into two steps. The first step, called Depth-wise, involves using a single filter to perform convolution on each depth layer of the input (if the input has 3 depth layers, there will be 3 filters in total). This step is used for feature extraction from the input. The second step, called Point-wise (with a kernel size of 1 × 1), combines the results from the Depth-wise step to create the desired number of filter outputs. FIGURE 2 shows a simple diagram of the Depth-wise Separable Convolution method. This method reduces the number of multiplications within the model, resulting in a smaller model size and lower computational resource usage, while maintaining accuracy. The concept can be explained as follows

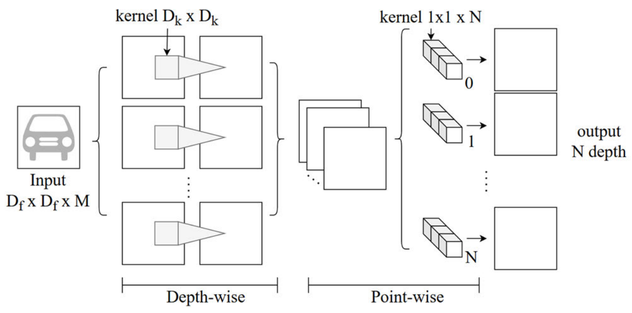


FIGURE 2. Illustrate the working mechanism of depth-wise separable convolution.

A standard convolution layer receives an input of size $D_f \times D_f \times M$ (where D_f is the width and height of the input image, M represents the number of depths) performs convolution with a filter kernel of size $D_k \times D_k \times N$ (where D_k is the width and height of the convolutional kernel, N represents the number of depths of kernel) and produces a feature map. The computational costs required are equal to

$$D_f \cdot D_f \cdot M \cdot D_k \cdot D_k \cdot N \tag{1}$$

When applying the Depth-wise Separable Convolution method to standard convolution, the computational costs required are equal to

$$D_f \cdot D_f \cdot M \cdot D_k \cdot D_k + D_f \cdot D_f \cdot M \cdot N \tag{2}$$

When compared, it can be observed that the computational resources required in equation (2) are less than in equation (1). An example of its application is MobileNet [15], where the research aims to create a CNN model that is not large, making it suitable for use on smartphones. The internal structure of the model applies the Depth-wise Separable Convolution method.

D. 1 × 1 CONVOLUTION DIMENSION REDUCTION

Another method to reduce computational resources, opposite to Depth-wise Separable Convolution, is to use a 1 × 1 convolution filter to reduce the size of the input matrix first (similar to the Point-wise method). After that, the convolution with the intended kernel size is performed. This method reduces the number of depth layers in the input, thus decreasing the number of calculations required. This is called Dimension Reduction. An example is as follows

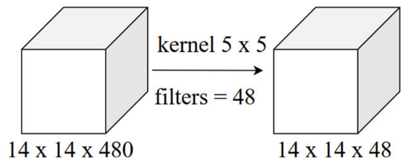


FIGURE 3. 5 × 5 general convolution.

FIGURE 3 shows an example of an input matrix of size 14 × 14 x 480. When performing calculations with 48 filters of 5 × 5 convolution, the total number of calculations will be (14 × 14×28) x (5 × 5×480) = 112.9 million calculations.

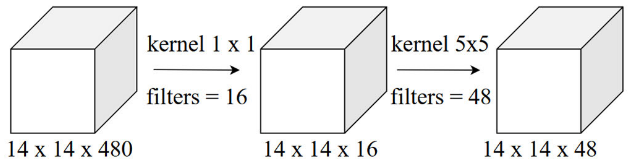


FIGURE 4. 5 × 5 convolution with Dimension reduction.

FIGURE 4: When using 1 × 1 convolution to reduce the size of the input matrix, the number of calculations is divided into two stages. The first stage, using 1 × 1 convolution, requires (14 × 14×16) x (1 × 1×480) = 1.5 million calculations. The

second stage, using 5×5 convolution, requires ($14 \times 14 \times 48$) x ($5 \times 5 \times 16$) = 3.8 million calculations. In total, the number of calculations is 1.5 million + 3.8 million = 5.3 million. Comparing the difference in the number of calculations, it is reduced by 112.9 million calculations.

The 1×1 convolution method became more well-known through GoogLeNet [25], also known as the Inception module (v1), which used 1×1 convolution to reduce the number of computations, helping to create a smaller model while maintaining similar classification accuracy. Another study, the Xception model [26], experimented with combining Depth-wise Separable Convolution with the Inception module that uses 1×1 convolution.

III. METHOD

This section proposes an image classification method for vehicle make classification. This part is divided into two parts. The former part discusses the structure of the CNN model designed for recognizing vehicle makes from front-view images, including a version of the model reduced in size using Depth-wise Separable Convolution and 1×1 Convolution Dimension Reduction. This is intended for use in processors with limited resources, such as embedded boards capable of processing CNN models, smartphones, or traffic cameras. Furthermore, the latter part covers the datasets, which was collected by our work. All images are long-range photographs of vehicle fronts that are captured at some toll plazas from free flow expressways in Thailand, as shown in FIGURE 5.

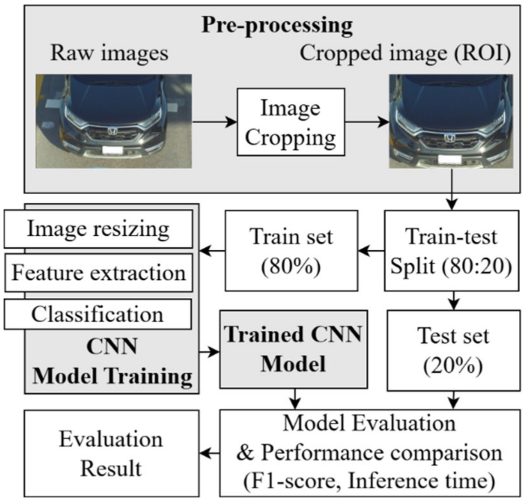


FIGURE 5. Overall experimental.

The structure of the experiment in this research for classifying vehicle make from front-view images of cars is as follows. First, explain the datasets, an overview, and how to collect them. Next, divide the process into two distinct tasks. The first is the pre-processing to crop the ROI [27] [28] vehicle image. The second is experimentation with the proposed CNN model.

A. DATASETS AND PRE-PROCESSING

In our study, the dataset was collected from images of cars with the same viewpoint as ones captured from free flow toll systems in Thailand. This dataset was self-collected in the same manner as other research projects which generally gathered their own datasets [29]. Since the free flow toll system aims to detect license plates, the dataset does not consist of full vehicle images but rather images of the front part of the car. Therefore, the images used in this study prominently feature the front of the car, including the hood, bumper, headlights, and the car logo. Recognizing the make and model of the car can be applied to toll systems, such as verifying car images without using license plates in cases where the license plate cannot be detected. To setup the viewpoint camera like the toll plazas, our camera was installed at a height of 5.3 meters from the road surface, with a detection range of approximately 7 meters, shown in FIGURE 6.

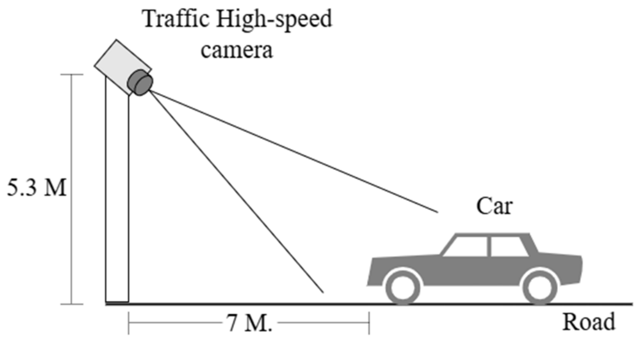


FIGURE 6. Installation and image capture setup using vehicle detection cameras.

The initial images captured by the vehicle detection camera are cropped using Yolo [30], [31] to isolate only the vehicle parts. It resulted in an image size of 244×244 pixels [32]. To ensure the dataset is balanced across all vehicle make classes, this study selected images from the entire dataset to represent five commonly found vehicle makes in Thailand: Toyota, Honda, Mazda, Nissan, and Isuzu. Each vehicle make has 200 images, so there are 1000 images in total. All images are captured during the daytime, as some examples are shown in FIGURE 7.

B. PROPOSED CCN MODEL

The proposed CNN model for vehicle make classification consists of three model types, focusing on enhancing accuracy and reducing complexity. Each model type was produced from a different filter block. The filter block includes convolutional layers, computing different styles to create a feature map.

1) FILTER-BLOCK

In this study, the Filter-boxes are designed, which is a group of processing layers that perform feature extraction. There are three types of Filter-Block: standard filter-box,

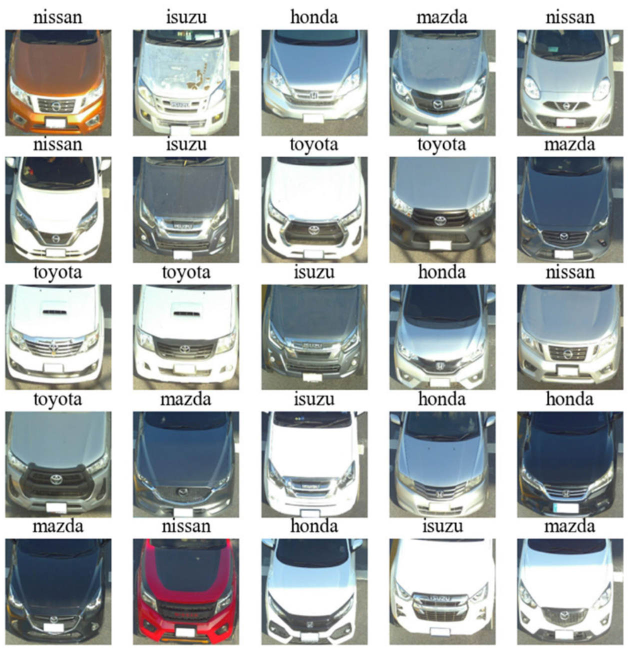


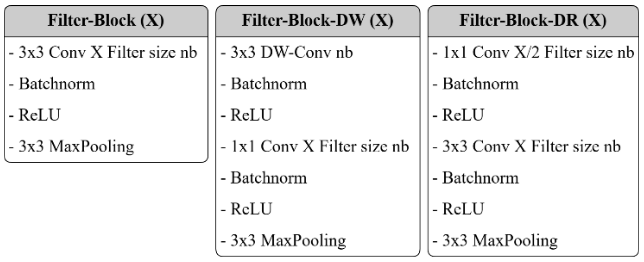
FIGURE 7. Examples of our dataset.

filter-block-DW applied depth-wise separatable convolutional method, and filter-block-DR applied dimension reduction method. The structures of all three types of Filter-Blocks are shown in FIGURE 8

First, **Standard Filter-Block** consists of an initial 3 × 3 convolution layer where the number of filters can be specified as needed, without using bias or activation function, with padding set to 'same' and stride = 1. This is followed by a batch normalization layer [33] to speed up the learning process, an activation function layer using ReLU, and finally, a 3 × 3 MaxPooling layer with stride = 2 (Overlap) [34]. Standard MaxPooling uses a stride equal to the kernel pooling size (in this case, stride = 3). Using a smaller stride for MaxPooling reduces the size of the output matrix.

Second, **Filter-Block-DW** applies the Depth-wise separable convolution method to the standard Filter-Block. The internal structure starts with a 3 × 3 depth-wise convolution, where the number of filters equals the depth of the input matrix, without using bias or activation function. This is followed by batch normalization and ReLU activation function layers. Next, a 1 × 1 convolution without bias and activation function is applied, where the number of output filters can be specified. This is followed by batch normalization and ReLU activation function layers, ending with MaxPooling with stride = 2.

Last, **Filter-Block-DR** uses the 1 × 1 convolution method to reduce the size of the input matrix in the standard Filter-Block. The internal structure starts with a 1 × 1 convolution with 16 filters, without using bias or activation function. This is followed by batch normalization and ReLU activation function layers. Next, a 3 × 3 convolution without bias



Filter-Block (X)	Filter-Block-DW (X)	Filter-Block-DR (X)
- 3x3 Conv X Filter size nb	- 3x3 DW-Conv nb	- 1x1 Conv X/2 Filter size nb
- Batchnorm	- Batchnorm	- Batchnorm
- ReLU	- ReLU	- ReLU
- 3x3 MaxPooling	- 1x1 Conv X Filter size nb	- 3x3 Conv X Filter size nb
	- Batchnorm	- Batchnorm
	- ReLU	- ReLU
	- 3x3 MaxPooling	- 3x3 MaxPooling

FIGURE 8. The three types of Filter-Blocks.

and activation function is applied, where the number of output filters can be specified. This is followed by batch normalization and ReLU activation function layers, ending with MaxPooling with stride.

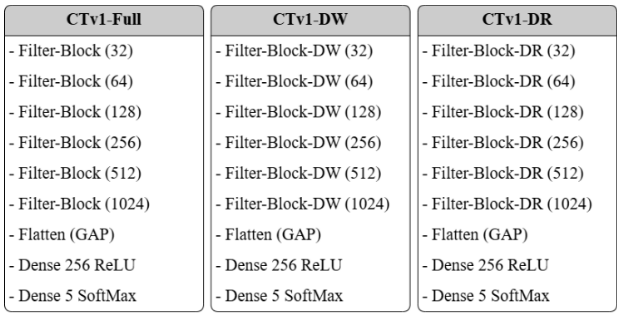
2) MODEL STRUCTURE

The previous subsection presented Filter-Block, Filter-Block-DW, and Filter-Block-DR. They have different internal operations, affecting the number of parameters and computational processes. After establishing the feature extraction using Filter-Blocks, a preliminary model structure was introduced. This structure, built from the three types of Filter-Blocks, is named "CTv1". The model structure starts from the top and progresses downward, it consists of six layers of Filter-Blocks. Each layer has a different number of output filters, arranged from fewer to more filters. This approach, which is adopted from ImageNet [1], increases the chances of detecting key features by reducing the matrix dimensions. The chosen numbers of six filters are 32, 64, 128, 512, and 1024 respectively. This format is used in MobileNet model architecture.

Model Algorithm

```
Input:    image_data //a vehicle RGB image sizing
            244*244*3
Output:   vm_pred //the prediction result of vehicle
  make
//Filter-Block
1:  Function Filter-Block(filter-size, map-in)
2:      m ← Multi-layer-filter(filter=filter-size, map-in)
//Model Structure
1:  map ← Filter-Block(32, image_data)
2:  map ← Filter-Block(64, map)
3:  map ← Filter-Block(128, map)
4:  map ← Filter-Block(256, map)
5:  map ← Filter-Block(512, map)
6:  map ← Filter-Block(1024, map)
7:  map ← Flatten_GAP(map)
8:  map ← Dense_256(map)
9:  map ← Dense_5(map)
10: vm_pred ← softmax(map)
```

After passing through all the Filter-Blocks, the model uses Global Average Pooling to flatten the 3D matrix into a 1D



CTv1-Full	CTv1-DW	CTv1-DR
- Filter-Block (32)	- Filter-Block-DW (32)	- Filter-Block-DR (32)
- Filter-Block (64)	- Filter-Block-DW (64)	- Filter-Block-DR (64)
- Filter-Block (128)	- Filter-Block-DW (128)	- Filter-Block-DR (128)
- Filter-Block (256)	- Filter-Block-DW (256)	- Filter-Block-DR (256)
- Filter-Block (512)	- Filter-Block-DW (512)	- Filter-Block-DR (512)
- Filter-Block (1024)	- Filter-Block-DW (1024)	- Filter-Block-DR (1024)
- Flatten (GAP)	- Flatten (GAP)	- Flatten (GAP)
- Dense 256 ReLU	- Dense 256 ReLU	- Dense 256 ReLU
- Dense 5 SoftMax	- Dense 5 SoftMax	- Dense 5 SoftMax

FIGURE 9. The three types of our models.

matrix. This is followed by a Dense layer with 256 units using the ReLU activator and a final Dense layer with 5 units using the SoftMax activator to predict the five vehicle brands. The other two models, which incorporate Filter-Block-DW and Filter-Block-DR, have structures similar to the CTv1 model, differing only in the Filter-Block sections. All other operational layers remain the same. The structures of all three model concepts are shown in the model pseudocode named "Model Algorithm" and FIGURE 9 following the pseudocode for the overall three model structures.

C. EXPERIMENTS AND EVALUATIONS

In the experiment, which randomly shuffled the data but set a seed value to ensure that the randomization results are consistent each time, allowing other models to receive the same randomization. Then split the data into training and testing sets, with 80% for training and 20% for testing, using the split_dataset function from the TensorFlow library [35].

To assess the experiment, the model size, parameter count, and execution time will be measured from the time the trained model is used, which is the inference time. Since the experiment involves multi-class classification, the F1-score was used to measure performance. The **F1-Score** is calculated from the **Recall** and **Precision** value. The Recall and Precision values come from the calculation of TP=True Positive, TN=True Negative, FP=False Positive, and FN=False Negative. The formula of the evaluation values follows.

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{3}$$

Precision: is the ratio of correctly predicted positive observations to the total predicted positive observation.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Recall: is the ratio of correctly predicted positive observations to the total positive observation class.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

IV. RESULTS

This section of the experiment will explain the initial preparation methods for both hardware and software. The experimental results focus on recognizing vehicle brands based on front-view images. The experiments used both state-of-the-art CNN models and the proposed models. The results will emphasize accuracy, model size, training and prediction time, and additional experiments involving parameter adjustments that affect the CNN model.

A. EXPERIMENTAL PREPARATION

In the experiment, the Python and machine learning libraries: TensorFlow from Google and Conda were used for managing the environment and installing various packages. For the vehicle brand classification experiment, all base models (VGG16 [35], InceptionV3 [9], ResNet50 [2], and MobileNet [15], Yolo11m-cls [31]) were used as pre-built models available in TensorFlow except for Yolo, which comes with tools from Ultralytics [31] and is trained from scratch. For the custom models, which were built entirely from scratch using the basic structure of CNNs. For training, the Adam optimizer and Sparse Categorical Cross-Entropy were used to calculate the loss for parameter adjustments within the model. In addition to software preparation, the necessary hardware for the experiment was also prepared. The summary of both hardware and software preparations is shown in TABLE 1.

TABLE 1. Hardware and software specification.

Topic	Spec
Processor	Intel i5-12400F
Graphical Processing Unit (GPU)	NVIDIA GeForce RTX 3060 ti
Memory	16 GB
Operating System	Windows 11
Python	3.9.18
TersonFlow	2.10
CudaToolkit	11.2
CuDNN	8.1

B. RESULT AND EVALUATION

In the experiments conducted on learning and testing car make recognition using CNN models, the well-known base models for classifying vehicle make and model [36], [37] were tested, which included VGG16 [35], InceptionV3 [9], ResNet50 [2], MobileNet [15], Yolo11m-cls [31] and the other three models, which are introduced in this paper, developed from three types of filter-blocks, resulting in three models: CTv1, CTv2-DW, and CTv2-DR. The experiments were conducted with a dataset of car images, with the number of learning epochs set to 200 (epoch = 200). The results were evaluated using F1 score and F1 weighted average as follows.

Model Performance of Vehicle Makes Classification As our experiments from well-known base models and our approaches, the performance estimation results using F1 w-avg (F1 weighted average) and F1 score as reported in TABLE 2. Some key highlights of the result are noticed as follows:

The CTv1 model took first place with the highest weighted average F1 score (0.99) and got a 1.00 F1 score for predicting

Honda and Isuzu, with Toyota being the lowest at 0.98. InceptionV3 came in second with the F1 w-avg score of 0.97, excelling in Mazda predictions (1.00) and scoring lowest for Nissan (0.92). With an F1 w-avg score of 0.93, the third-ranked CTv1-DR outperformed both Nissan (0.88) and Mazda (0.98).

Fourth place went to Yolo11m-cls, scoring 0.92 F1 w-avg, with a peak at Honda 1.00 and a lowest at Nissan 0.85. With an F1 w-avg of 0.90, CTv1-DW came in fifth place. Honda's peak score was 0.93, while Nissan's was 0.86. ResNet50 ranked sixth, with the F1 w-avg of 0.81, highest for Mazda (0.87) and lowest for Nissan (0.70). MobileNet was seventh with the F1 w-avg of 0.65, excelling at 0.76 for Mazda and scoring lowest for Nissan at 0.39. Lastly, VGG16 placed seventh with the lowest F1 w-avg (0.23), predicting only Toyota with an F1 score of 0.38. Figure 10 presents examples of classification results from the CTv1 model, which include the probability scores for each class.

TABLE 2. F1 scores and average F1 of the vehicle makes classification.

Model	F1 Scores of each Make					F1 Weighted Average
	Honda	Isuzu	Mazda	Nissan	Toyota	
VGG16 [35]	0.00	0.00	0.00	0.00	0.38	0.23
InceptionV3 [9]	0.99	0.98	1.00	0.92	0.96	0.97
ResNet50 [2]	0.85	0.83	0.87	0.70	0.79	0.81
MobileNet [15]	0.70	0.68	0.76	0.39	0.64	0.65
Yolo11m-cls [31]	1.00	0.95	0.90	0.85	0.90	0.92
CTv1	**1.00**	**1.00**	**0.99**	**0.99**	**0.98**	**0.99**
CTv1-DW	0.93	0.87	0.92	0.86	0.91	0.90
CTv1-DR	0.95	0.90	0.98	0.88	0.92	0.93

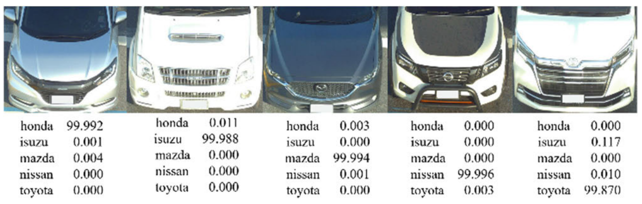


FIGURE 10. The classification result example includes the probability scores for each class.

1) COMPUTATION RESOURCE FOR CLASSIFICATION MODELS

In addition to the model performance, the computation resources of each model are also evaluated in TABLE 2. The inference time (I-Time), and the average inference time of each model resulted in Table 4.3. The experiment by timing the models using 200 front-view car images from the test set was conducted, with each model performing calculations 8 times, 25 images per run. The baseline VGG16 model had the largest size with 134.28M parameters, followed by ResNet50, InceptionV3, Yolo11m-cls and MobileNet, respectively. The proposed models by the researchers, CTv1, had the largest size with 6.5M parameters, followed by CTv1-DW and CTv1-DR.

2) INFEENCE TIME EVALUATED BY A RESOURCE-CONSTRAIN DEVICE

Since this work also aims to create a vehicle make classification model for embedding in a portable device having lower computational resources, the execution time of the proposed model (CTv1) and baseline models VGG16, InceptionV3, Yolo11m-cls, ResNet50, and MobileNet will be measured on an ARM-based Raspberry Pi 3 Model B board. The board has a Broadcom BCM2837B0 SoC with a 1.4GHz quad-core A53 (ARMv8) 64-bit processor and 1GB LPDDR2 SDRAM, which makes it a device with limited resources. The objective of the research was to develop a model that could operate on such devices. The test results indicated that the CTv1 model had an average execution time of about 1 second, similar to MobileNet's average time of 0.99 seconds, and the CTv1 had a power consumption of 25 mWh(milliwatts-hour), while MobileNet had 38 mWh. InceptionV3 had a power consumption of 90 mWh, Yolo11m-cls of 126 mWh, and ResNet50 of 103 mWh, with each having an average time exceeding 5 seconds. The VGG16 model could not run on the Raspberry Pi 3 Model B due to insufficient memory as indicated by the operating system.

V. DISCUSSION

From the experiments using state-of-the-art CNN models (VGG16 [35], InceptionV3 [9], ResNet50 [2], MobileNet [15], Yolo11m-cls [31]) and the proposed CNN models (CTv1, CTv1-DW, and CTv1-DR) for classifying car makes from front-view images, the results showed a relationship between model size and training time. This section will discuss the model architecture, its contribution, its application conditions, and the opportunities for future research.

A. MODEL ARCHITECTURE

1) MODEL SIZE

As reviewing the result, larger models took more time to train, but they did not always result in better prediction accuracy (as seen in the previous results in TABLE 2 for F1 w-avg and F1 score). TABLE 2 shows the model sizes and training times for 200 epochs. VGG16 [18] had the largest model size at 134.28M total parameters and the longest training time but had the lowest F1 w-avg and F1 score. InceptionV3 [4] had the best F1 w-avg and F1 score among the base models, with a size of 23.59M total parameters and a training time of 945.07 seconds. CTv1 had the highest F1 w-avg and F1 score, with a size and training time about three times less than InceptionV3 [4]. CTv1-DW was about 6.5 times smaller than CTv1 but had about 9% lower accuracy, and CTv1-DR was about 11 times smaller than CTv1 but its performance gave a little bit lower as about 6%. The DW and DR models are smaller because the size of the kernel (the number of kernel parameters) is reduced compared to

TABLE 3. Model size and Training - Inference time.

Model	Computation Time (seconds)		Model Size (Number of Parameters)		
	Training	Inference	Trainable Parameters	Non-Trainable Parameters	Total Parameters
VGG16 [35]	2,045.67	0.275714	134,281,029	0	134,281,029
InceptionV3 [9]	945.07	0.127029	21,778,597	34,432	21,813,029
ResNet50 [2]	1,204.29	0.056198	23,544,837	53,120	23,597,957
MobileNet [15]	427.71	0.052765	3,212,101	21,888	3,233,989
Yolo11m-cls [31]	734.43	0.125102	10,348,037	0	10,348,037
CTv1	354.96	0.037517	6,551,877	4,032	6,555,909
CTv1-DW	418.04	**0.012108**	974,115	6,022	980,137
CTv1-DR	435.59	0.013200	572,021	4,224	**576,245**

TABLE 4. The prediction time when tested on a Raspberry Pi 3 Model B.

Model	**Inference Time** (average seconds per image)
VGG16 [35]	Not enough memory to run
InceptionV3 [9]	5.23
ResNet50 [2]	5.46
MobileNet [15]	0.99
Yolo11m-cls [31]	5.26
CTv1	1.02
CTv1-DW	**0.78**
CTv1-DR	0.92

TABLE 5. The power consumption on a Raspberry Pi 3 Model B.

Model	Power consumption (mWh)	Peak Power (W)
InceptionV3 [9]	90	4.43
ResNet50 [2]	103	4.42
MobileNet [15]	38	4.23
Yolo11m-cls [31]	126	4.42
CTv1	25	4.22
CTv1-DW	20	4.07
CTv1-DR	21	4.29

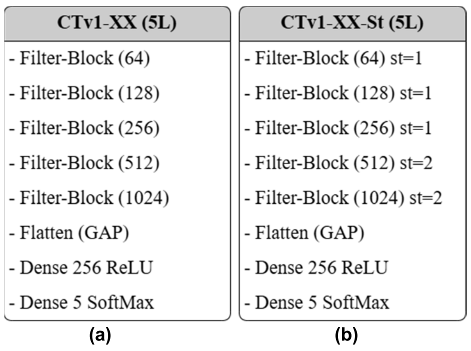
calculations using a standard kernel, resulting in an overall smaller model size. However, the accuracy of the results does not decrease significantly. Based on analysis and testing, this study realizes that the parameters in the kernel that are removed in the DW or DR calculations are less important than the remaining parameters.

There are other methods for reducing the size of CNN models, such as pruning. Preliminary studies [38] have indicated that it is possible to reduce the size of the model while maintaining its classification ability, with only a slight decrease in accuracy. This study started with creating a new one and found good results, which meant that pruning was not within the scope of this research. However, thank you to the reviewer for bringing this issue up again and the experiment in feature work with the proposed model.

2) MODIFYING PADDING AND STRIDE

In the previous main experiment, the proposed small CNN model is aimed at use in resource-constrained processing devices. The model size directly affects the memory requirements of the device, but the complexity of processing also directly impacts processing time. Although the CTv1 model could operate on the Raspberry Pi 3 Model B with an average processing speed of 1 second per image, the padding and stride parameters were adjusted in the convolutional layer. These parameters do not affect the model size but do impact processing complexity. Therefore, additional experiments were conducted by creating the CTv1-XX-NP model, which includes versions of CTv1, CTv1-DW, and CTv1-DR without padding in the filter-block layer that is a convolution layer. This adjustment resulted in a slight reduction in the size of the feature map output at each layer after passing through the convolution layer. Consequently, the original CTv1 model, which had 6 filter-block layers, could not process at the 1024 layer because the matrix from the 512 layer was $4 \times 4 \times 512$. Therefore, the first 32 filter-blocks were removed.

CTv1-XX-St, in FIGURE 12(b-a), adjusts the CTv1, CTv1-DW, and CTv1-DR models by changing the convolution layer's stride from the default value of 1, where the kernel overlaps during feature extraction, to a stride of 2 in some layers. When using a 3×3 kernel, this prevents overlap during feature extraction. Increasing the stride reduces the size of the feature map output after passing through the convolution layer, so the number of Filter-Block layers is reduced to 5.



CTv1-XX (5L)	CTv1-XX-St (5L)
- Filter-Block (64)	- Filter-Block (64) st=1
- Filter-Block (128)	- Filter-Block (128) st=1
- Filter-Block (256)	- Filter-Block (256) st=1
- Filter-Block (512)	- Filter-Block (512) st=2
- Filter-Block (1024)	- Filter-Block (1024) st=2
- Flatten (GAP)	- Flatten (GAP)
- Dense 256 ReLU	- Dense 256 ReLU
- Dense 5 SoftMax	- Dense 5 SoftMax
(a)	(b)

FIGURE 11. Model structure No padding (a) and Stride (b).

Adjusting the padding and stride does not affect the number of parameters within the model because these changes only alter the kernel's operational boundaries. The number of parameters in the kernel remains the same, but the resulting feature map matrix size differs with different padding and stride values. For example, the training and inference times are ordered as CTv1-St < CTv1-NP < CTv1, which corresponds to the feature map matrix size during operation; CTv1-St has the smallest matrix size, followed by CTv1-NP, and finally CTv1 with the largest. The accuracy in vehicle brand recognition shows that CTv1 and CTv1-NP have similar values, while CTv1-St is slightly lower.

TABLE 6. F1 score and F1 weighted average of the vehicle makes classification (Padding, Stride).

Model	F1 Scores of each Make					F1 Weighted Average
	Honda	Isuzu	Mazda	Nissan	Toyota	
CTv1 (5L)	**0.98**	**1.00**	**1.00**	**0.96**	**0.97**	**0.98**
CTv1-DW (5L)	0.92	0.88	0.96	0.83	0.88	0.90
CTv1-DR (5L)	0.93	0.97	0.94	0.89	0.92	0.93
CTv1-NP (5L)	**1.00**	**1.00**	**0.99**	**0.97**	**0.97**	**0.98**
CTv1-DW-NP (5L)	0.95	0.92	0.92	0.91	0.94	0.93
CTv1-DR-NP (5L)	0.93	0.91	0.98	0.89	0.92	0.93
CTv1-St (5L)	0.99	0.92	0.98	0.88	0.96	0.95
CTv1-DW-St (5L)	0.81	0.74	0.91	0.70	0.74	0.78
CTv1-DR-St (5L)	0.93	0.89	0.95	0.82	0.93	0.91

3) BATCH NORMALIZATION

In the structure of the models presented, CTv1, CTv1-DW, and CTv1-DR, which include Filter-blocks with Batch-normalization (batchnorm) layers, it was found through experiments that using Batch normalization helps both in terms of classification accuracy and the time taken for model training. To demonstrate the differences, models with and without batch normalization were tested. The results are shown in TABLE 8. From TABLE 8, in the CTv1 model without batchnorm, the F1-weight average decreased from 0.99 to 0.76, which is approximately a 25% reduction, and the training time increased to 3,354.51 seconds, nearly a tenfold increase.

The model CTv1-DW without batchnorm, the F1-weight average dropped from 0.90 to 0.74, by approximately 16%, and the training time increased to 1,511.66 seconds from the original 418.04, an increase of about 3.5 times. Finally, the CTv1-DR model's F1-weight average decreased from 0.93 to 0.16, a reduction of more than 80%, and the training time taken increased from 435.59 seconds to 2,113.53 seconds, an increase of approximately 4.8 times.

4) CONTRIBUTION

A compact CNN model for car make recognition is presented in this work, which maximizes computational efficiency and memory utilization. The model employs more filters by decreasing the feature map dimension and increasing the network depth, which produces richer feature extraction while keeping the model size minimal. This makes it the perfect and affordable option for devices with limited resources, such as tolling systems or traffic cameras.

Our tests demonstrate that accuracy is not necessarily increased by larger models. For example, VGG16 had the lowest F1 scores even though it had the highest size (134.28M parameters). Among base models, InceptionV3 had the highest accuracy with 23.59M parameters. With a three-fold reduction in both model size and training time, the suggested CTv1 model outperformed InceptionV3, and CTv1-DW and CTv1-DR further decreased model size without appreciably lowering accuracy (by 9% and 6%, respectively). Depth-wise separable convolutions were used to accomplish these reductions, demonstrating that performance may be preserved while model size is reduced by eliminating less important kernel parameters.

5) APPLICABLE CONDITIONS

In testing the robustness of the presented CNN model, the researchers simulated scenarios similar to camera obstructions, such as blur, noise, lighting, and water drops. This was conducted on initial images with a size of approximately 600×600 pixels before testing with the CTv1 model as shown in FIGURE 13, the test results are shown in FIGURE 14, and the discussion is as follows.

The results of the experiment simulating water drop occlusion in front of the camera (waterdrop) involved simulating 2 to 11 water drops with random sizes ranging from 10 to 30 pixels on the test set image. The experimental results show that in the water drop scenario, the CTv1 model still has an F1 weight average of approximately 0.98 - 0.99, which is close to the normal system. It is predicted that the water drops on the camera lens do not significantly obscure various parts of the image, allowing the overall data or important points to remain intact. Therefore, the impact on the classification of the CTv1 model is minimal.

The testing involved adding noise to the images. Two types of noise were tested: the first was Gaussian Noise, which was tested with a mean of 0 and a standard deviation adjusted from 10 to 100. This resulted in the F1 weight average of the CTv1 model decreasing from 0.99 to approximately 0.70. Analyzing the images with added Gaussian Noise showed that the color values in some positions were altered, which affected the calculation of the feature map during the convolutional process. This was particularly evident when testing with salt and pepper noise, which randomly adds white or black color values to the image based on a defined probability. The test results graph indicated a linear trend of the F1 weight average decreasing. The nature of salt and pepper noise involves adding white and black colors, and the white color, when processed in the convolutional process, has a chance of being considered an important feature, affecting the classification.

As for the testing when the image was blurred, the blur was tested by adjusting the size of the blurring kernel from 1×1 (which means no blur) to 10×10. It was observed that

TABLE 7. Model size and Training Time (Padding, Stride).

Model	Computation Time (seconds)		Model Size (Number of Parameters)		
	Training	Inference	Trainable Parameters	Non-Trainable Parameters	Total Parameters
CTv1 (5L)	720.93	0.039764	6,534,277	3,968	6,538,245
CTv1-DW (5L)	674.12	0.022632	971,811	5,894	977,705
CTv1-DR (5L)	805.04	0.033189	566,853	4,128	570,981
CTv1-NP (5L)	631.43	0.032168	6,534,277	3,968	6,538,245
CTv1-DW-NP (5L)	626.31	0.021560	971,811	5,894	977,705
CTv1-DR-NP (5L)	726.89	0.030220	566,853	4,128	570,981
CTv1-St (5L)	607.22	0.029466	6,534,277	3,968	6,538,245
CTv1-DW-St (5L)	622.43	0.014284	971,811	5,894	977,705
CTv1-DR-St (5L)	720.71	0.030759	566,853	4,128	570,981

TABLE 8. Compare the F1-weight average and training time between the model with a batch normalization layer and the model without a batch normalization layer.

Model	With Batch-normalization	
	F1-Weight Average	Training Time(s)
CTv1	0.99	354.96
CTv1-DW	0.90	418.04
CTv1-DR	0.93	435.59
	Without Batch-normalization	
CTv1	0.76	3,354.51
CTv1-DW	0.74	1,511.66
CTv1-DR	0.16	2,113.53

in the kernel size range of 1-5, the F1 weight average value decreased slightly, but after that, it decreased significantly. From the analysis, it is expected that blurring the image causes important information or features in the image to disappear. If the blur does not remove important information or features in the image, the model can still classify with reduced accuracy. However, if the blur causes important characteristics or information in the image to decrease or disappear, it will affect the model's classification.

Finally, the test to reduce the brightness of the image was conducted by adjusting the brightness level from 0 to 100. From the tests, it was shown that the CTv1 model had a decreasing F1 weight average according to the level of brightness reduction. Although the logo of the vehicle make was still somewhat visible to the naked eye, the analysis indicated that the CTv1 model also used other components in conjunction with the vehicle logo to classify the vehicle make. Therefore, reducing the image brightness affected the F1 weight average of the model.

6) OPPORTUNITIES FOR FUTURE RESEARCH

Even though the research's images are clear and unobstructed, there may be instances in which dust on the camera lens, paper, leaves, insects, or water droplets could block the view. These elements might make the suggested methods less effective. Therefore, in order to test and improve the robustness of the model, future work might entail simulating such scenarios, much like the study in [39]. The system could also be further enhanced by investigating different feature descriptors [22], concentrating on the thorough classification of particular components [40], or putting attention filters in place filter [41], [42], [43], [44]. In the future, the work intends to create CNN models to categorize different kinds of vehicles in peculiar circumstances.

Using vehicle classes that are frequently found in Thailand, the demand for the country's free-flow tolling system was evaluated in this study. Our study focuses on these classes because five particular brands are very well-liked. Our architecture can be extended for further training if enough information on other car brands becomes available in the future.

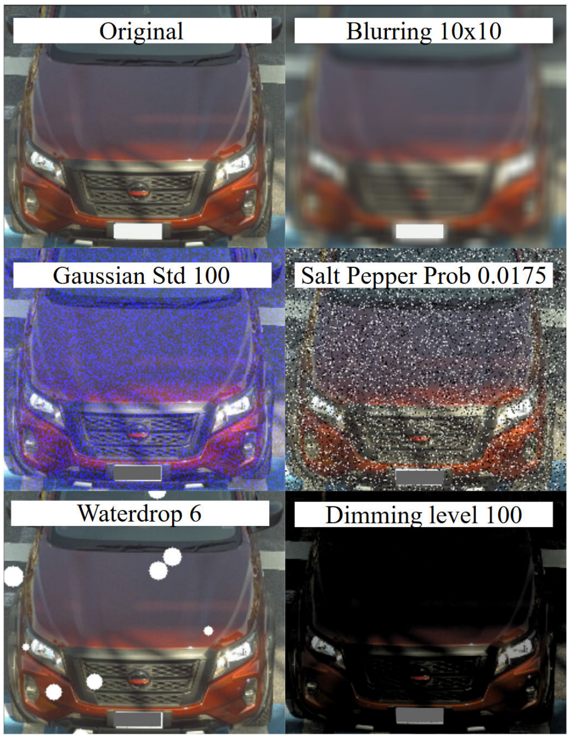


FIGURE 12. The CTv1 model's robustness is tested using a simple image that incorporates blurring, Gaussian noise, salt and pepper noise, waterdrop simulation, and light dimming.

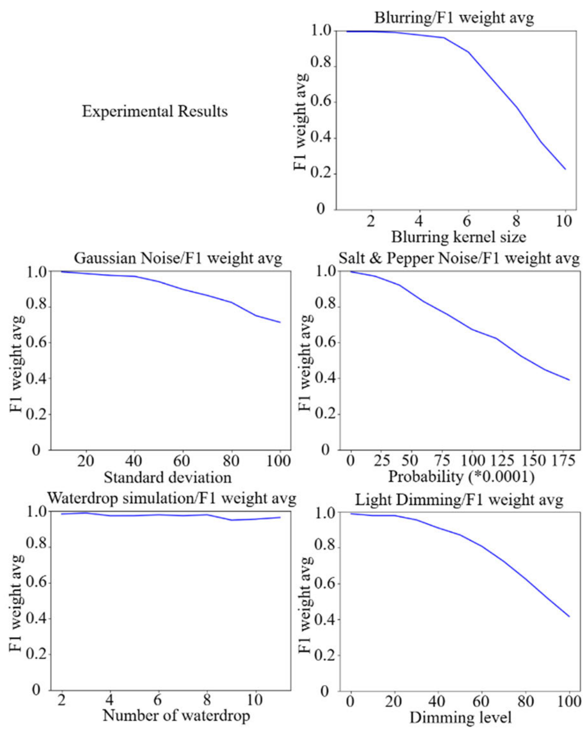


FIGURE 13. The result of testing the CTv1 model's robustness contains blurring, Gaussian noise, salt and pepper noise, waterdrop simulation, and light dimming compared with the F1-weight average.

This study's vehicle detection method makes use of the YOLO model, which has a high memory and processing overhead. Future studies will look into developing a more effective vehicle detection model that can be used with low-resource hardware.

VI. CONCLUSION

From the study and development of a small Convolutional Neural Network (CNN) model for use in resource-constrained processing devices to recognize car makes from frontal images, this study developed a CNN model with 99% accuracy in recognizing car makes. This model is smaller than the state-of-the-art CNN models tested, which achieved over 90% accuracy (VGG16, InceptionV3, Yolo11m-cls, and ResNet50). The proposed model achieved 99% accuracy with shorter training and prediction times. It was tested on a resource-limited device, the Raspberry Pi 3 Model B, where the CTv1 model operated with an average processing time of 1 second per image and a power consumption of 20 mWh.

The model uses a simple CNN structure and reduces the dimension of feature maps while increasing the number of layers. Testing with Depth-wise Separable Convolution reduced the number of parameters but also decreased accuracy. The 1×1 Convolutional Dimension Reduction (Bottleneck) method can reduce size if the output filter count is not too high, but it increased training and prediction times compared to Depth-wise Separable Convolution. Not using padding in the convolution layer did not affect learning accuracy but limited network construction due to reduced feature map size after each convolution layer. Stride also reduced feature map size but had a greater negative impact on prediction accuracy.

Moreover, the robustness testing was additionally conducted. The issues of blurring, Gaussian noise, salt and pepper noise, waterdrop simulation, and light dimming were simulated and tested. In this regard, increasing the level of noise gradually highlights the robustness boundaries of the model more clearly. This is an intriguing aspect that can be further explored for improving the CNN model as well as studying appropriate preprocessing techniques and pruning methods to enhance vehicle model classification in any challenging conditions.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 60, May 2017, pp. 84–90.

[2] X. Ma and A. Boukerche, "An AI-based visual attention model for vehicle make and model recognition," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6.

[3] A. Nazemi, Z. Azimifar, M. J. Shafiee, and A. Wong, "Real-time vehicle make and model recognition using unsupervised feature learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3080–3090, Jul. 2020.

[4] S. H. Tan, J. H. Chuah, C.-O. Chow, J. Kanesan, and H. Y. Leong, "Artificial intelligent systems for vehicle classification: A survey," *Eng. Appl. Artif. Intell.*, vol. 129, Mar. 2024, Art. no. 107497.

[5] G. Doğan and B. Ergen, "A new approach based on convolutional neural network and feature selection for recognizing vehicle types," *Iran J. Comput. Sci.*, vol. 6, no. 2, pp. 95–105, Jun. 2023.

[6] W. Pan, X. Zhou, T. Zhou, and Y. Chen, "Fake license plate recognition in surveillance videos," *Signal, Image Video Process.*, vol. 17, no. 4, pp. 937–945, Jun. 2023.

[7] S. Naseer, S. M. A. Shah, S. Aziz, M. U. Khan, and K. Iqtidar, "Vehicle make and model recognition using deep transfer learning and support vector machines," in *Proc. IEEE 23rd Int. Multitopic Conf. (INMIC)*, Nov. 2020, pp. 1–6.

[8] D. Liu and Y. Wang, "Monza: Image classification of vehicle make and model using convolutional neural networks and transfer learning," Stanford Univ., Stanford, CA, USA, Tech. Rep., 2017. [Online]. Available: https://scholar.google.com/scholar?cluster=8348522735565363626

[9] B. Balci, A. Elihos, M. Turan, B. Alkan, and Y. Artan, "Front-view vehicle make and model recognition on night-time NIR camera images," in *Proc. 16th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Sep. 2019, pp. 1–6.

[10] B. Balci and Y. Artan, "Few-shot learning for vehicle make & model recognition: Weight imprinting vs. nearest class mean classifiers," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

[11] A. Boukerche and X. Ma, "A novel smart lightweight visual attention model for fine-grained vehicle recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13846–13862, Aug. 2022.

[12] A. J. Siddiqui and A. Boukerche, "A novel lightweight defense method against adversarial patches-based attacks on automated vehicle make and model recognition systems," *J. Netw. Syst. Manage.*, vol. 29, no. 4, p. 41, Oct. 2021.

[13] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015, *arXiv:1511.08458*.

[14] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," 2014, *arXiv:1403.1687*.

[15] 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 (2017)," 2017, *arXiv:1704.04861*.

[16] K. Siu, D. M. Stuart, M. Mahmoud, and A. Moshovos, "Memory requirements for convolutional neural network hardware accelerators," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2018, pp. 111–121.

[17] A. Boukerche and X. Ma, "Vision-based autonomous vehicle recognition: A new challenge for deep learning-based systems," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–37, May 2022.
[18] T. Ghosh, S. Gayen, S. Maity, D. Valenkova, and R. Sarkar, "A feature fusion based custom deep learning model for vehicle make and model recognition," in *Proc. 13th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2024, pp. 1–4.
[19] S. H. Tan, J. H. Chuah, C.-O. Chow, and J. Kanesan, "Coarse-to-fine context aggregation network for vehicle make and model recognition," *IEEE Access*, vol. 11, pp. 126733–126747, 2023.
[20] S. Hayee, F. Hussain, and M. H. Yousaf, "A novel FDLSR-based technique for view-independent vehicle make and model recognition," *Sensors*, vol. 23, no. 18, p. 7920, Sep. 2023.
[21] S. Nanayakkara and R. G. N. Meegama, "A deep learning approach for scoring quality in vehicle grading," *Eng. Appl. Artif. Intell.*, vol. 131, May 2024, Art. no. 107812.
[22] S. Jabeen, A. Jabeen, S. M. Adnan, and W. A. Rao, "Vehicle make and model recognition using structural and pattern based feature descriptors," in *Proc. Int. Conf. Commun. Technol. (ComTech)*, Sep. 2021, pp. 73–78.
[23] H. C. Sánchez, N. H. Parra, I. P. Alonso, E. Nebot, and D. Fernández-Llorca, "Are we ready for accurate and unbiased fine-grained vehicle classification in realistic environments?" *IEEE Access*, vol. 9, pp. 116338–116355, 2021.
[24] S. Yang, J. Zhang, C. Bo, M. Wang, and L. Chen, "Fast vehicle logo detection in complex scenes," *Opt. Laser Technol.*, vol. 110, pp. 196–201, Feb. 2019.
[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[26] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[27] A. Amirkhani and A. H. Barshooi, "DeepCar 5.0: Vehicle make and model recognition under challenging conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 541–553, Jan. 2023.
[28] L. Lu and H. Huang, "Component-based feature extraction and representation schemes for vehicle make and model recognition," *Neurocomputing*, vol. 372, pp. 92–99, Jan. 2020.
[29] Y. Lyu, I. Schiopu, B. Cornelis, and A. Munteanu, "Framework for vehicle make and model recognition—A new large-scale dataset and an efficient two-branch-two-stage deep learning architecture," *Sensors*, vol. 22, no. 21, p. 8439, Nov. 2022.
[30] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with YOLOv8," 2023, *arXiv:2305.09972*.
[31] G. Jocher and J. Qiu, "Ultralytics YOLO11," GitHub, San Francisco, CA, USA, Tech. Rep., 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11/#citations-and-acknowledgements
[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.
[34] C. Kong and S. Lucey, "Take it in your stride: Do we need striding in CNNs?" 2017, *arXiv:1712.02502*.
[35] D. Avianto, A. Harjoko, and Afiahayati, "CNN-based classification for highly similar vehicle model using multi-task learning," *J. Imag.*, vol. 8, no. 11, p. 293, Oct. 2022.
[36] Y. Yu, L. Xu, W. Jia, W. Zhu, Y. Fu, and Q. Lu, "CAM: A fine-grained vehicle model recognition method based on visual attention model," *Image Vis. Comput.*, vol. 104, Dec. 2020, Art. no. 104027.
[37] S. Ghassemi, A. Fiandrotti, E. Caimotti, G. Francini, and E. Magli, "Vehicle joint make and model recognition with multiscale attention windows," *Signal Process., Image Commun.*, vol. 72, pp. 69–79, Mar. 2019.
[38] D. Tian, S. Yamagiwa, and K. Wada, "Heuristic method for minimizing model size of CNN by combining multiple pruning techniques," *Sensors*, vol. 22, no. 15, p. 5874, Aug. 2022.
[39] J. Ding, L. Linze, L. Rongchang, W. Cong, X. Tianyang, and X.-J. Wu, "Robust person re-identification approach with deep learning and optimized feature extraction," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW), IEEE*, Jul. 2024, pp. 1–6.
[40] M. Bularz, K. Przystalski, and M. Ogorzałek, "Car make and model recognition system using rear-lamp features and convolutional neural networks," *Multimedia Tools Appl.*, vol. 83, no. 2, pp. 4151–4165, Jan. 2024.
[41] M. Kanwal, M. M. Riaz, S. S. Ali, and A. Ghafoor, "Image attention retargeting using defocus map and bilateral filter," *Multimedia Tools Appl.*, vol. 79, nos. 27–28, pp. 19063–19073, Jul. 2020.
[42] M. Kanwal, M. M. Riaz, S. S. Ali, and A. Ghafoor, "Fusing color, depth and histogram maps for saliency detection," *Multimedia Tools Appl.*, vol. 81, no. 12, pp. 16243–16253, May 2022.
[43] M. Kanwal, M. M. Riaz, S. S. Ali, and A. Ghafoor, "Saliency-based fabric defect detection via bag-of-words model," *Signal, Image Video Process.*, vol. 17, no. 4, pp. 1687–1693, Jun. 2023.
[44] M. Kanwal, M. M. Riaz, and A. Ghafoor, "Unveiling underwater structures: Pyramid saliency detection via homomorphic filtering," *Multimedia Tools Appl.*, vol. 83, no. 29, pp. 72727–72744, Feb. 2024.
[45] A. A. Jamil, F. Hussain, M. H. Yousaf, A. M. Butt, and S. A. Velastin, "Vehicle make and model recognition using bag of expressions," *Sensors*, vol. 20, no. 4, p. 1033, Feb. 2020.

WIPUT PUISAMLEE received the B.Eng. degree in computer engineering from the Rajamangala University of Technology Phra Nakhon, Bangkok, Thailand, in 2013, and the M.Eng. degree in computer engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, in 2018.

From 2013 to 2018, he was a Researcher with the Embedded System Laboratory, King Mongkut's Institute of Technology Ladkrabang. From 2018 to 2021, he was a Software and Hardware Engineer with Linkflow Company Ltd., Thailand. From 2021 to 2022, he was a Staff Member with the Strategy and Planning Department, Rajamangala University of Technology Phra Nakhon. He is currently a Lecturer with the Department of Computer Engineering, Rajamangala University of Technology Phra Nakhon.

RATHACHAI CHAWUTHAI received the B.Eng. degree in computer engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 2006, the M.Eng. degree in information management from Asian Institute of Technology, Pathum Thani, Thailand, in 2012, and the Ph.D. degree in informatics from SOKENDAI, Kanagawa, Japan, in 2016. From 2006 to 2010, he was a Software Engineer with Thomson Reuters, Thailand. From 2012 to 2013, he was a Senior Software Engineer with Punsarn Asia, Thailand. From 2013 to 2016, he was a Research Assistant with the National Institute of Informatics, Japan. He is currently an Associate Professor with the Department of Computer Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang.