Current Applied Science and Technology Vol. 22 No. 1 (January-February 2022)

Research article

Fast and Accurate Deep Learning Architecture on Vehicle Type Recognition

Narong Boonsirisumpun and Olarik Surinta*

Multi-agent Intelligent Simulation Laboratory (MISL), Department of Information Technology, Faculty of Informatics, Mahasarakham University, Thailand

Received: 15 November 2020, Revised: 11 April 2021, Accepted: 18 May 2021

DOI: 10.55003/cast.2022.01.22.001

Abstract

Keywords

Vehicle Type Image Recognition;

image classification;

Convolutional Neural Network;

deep learning;

pattern recognition;

image recognition

Vehicle Type Recognition has a significant problem that happens when people need to search for vehicle data from a video surveillance system at a time when a license plate does not appear in the image. This paper proposes to solve this problem with a deep learning technique called Convolutional Neural Network (CNN), which is one of the latest advanced machine learning techniques. In the experiments, researchers collected two datasets of Vehicle Type Image Data (VTID I & II), which contained 1,310 and 4,356 images, respectively. The first experiment was performed with 5 CNN architectures (MobileNets, VGG16, VGG19, Inception V3, and Inception V4), and the second experiment with another 5 CNNs (MobileNetV2, ResNet50, Inception ResNet V2, Darknet-19, and Darknet-53) including several data augmentation methods. The results showed that MobileNets, when combine with the brightness augmented method, significantly outperformed other CNN architectures, producing the highest accuracy rate at 95.46%. It was also the fastest model when compared to other CNN networks.

1. Introduction

The increasing traffic problem around the world has forced a number of countries to build a system for assisting officers in controlling and securing their transportation problems. One of those countries is Thailand. Every year, more cars on the road are increasing the complexity of transportation control for the Thai government. Many researchers have tried to solve this problem for several years by adding a street video surveillance system to help police officers capture the illegal use of vehicles without them having to stand on the road, as happened in the past.

*Corresponding author: Tel.: (+66) 43-754359
E-mail: olarik.s@msu.ac.th

The key problem of such a system is how to detect a vehicle position in the image and how to extract the vehicle detail out from its license plate. But in many situations, the license plate does not appear clearly in the scene. That forces the system to identify other details of the vehicle instead, and included are vehicle shape, vehicle model, and vehicle type (car, truck, or bus) (see Figure 1). If the system can recognize these kinds of data, it helps to reduce the recognition processes on that vehicle. Many researchers brought these issues into the fields of image processing and machine learning to find a solution. Many state-of-the-art techniques have been applied to solve this problem, for example, Harris Corner Detector [1], Histogram of Oriented Gradients [2], or K-nearest neighbor [3]. But these techniques require a long time for the training process and are not accurate enough.

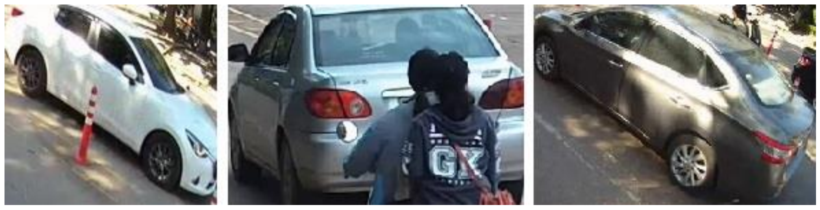

Figure 1. Vehicle image without license plate detail

Fortunately, the development of another technique called "the Convolutional Neural Network" or "CNN", which is based on the deep neural network structure, has been proved to solve the problem. Dong *et al*. [4], Huttunen *et al*. [5], and Bautista *et al*. [6] found that CNN achieved better performance on vehicle type recognition problems when compared to previous techniques. But there still was a limit when the vehicle image came from a different viewing angle or direction. These problems need to be fixed with an improved CNN or other additional technique.

The first modern model of CNN was AlexNet [7], which proved that the idea of CNN was a better method to solve various image recognition problems including problems with the vehicle image. After AlexNet, other CNN models were developed, for example, VGGNet [8], GoogLeNet [9], and MobileNets [10], and they showed more signs of progress. Each architecture enhanced the performance of CNN in various ways. They demonstrated improved recognition accuracy, network structure, or other model parameters. Several key problems with image recognition were solved with these developments [11].

Even though CNN has been a success in various areas of image recognition, they still face significant accuracy problems if the number of training datasets is so small or if the quality of the raw image is not satisfactory. Then, the addition of data augmentation techniques requires the creation of more variance in the dataset and helps to improve the performance. From all the problems mentioned above, the application of the CNN algorithms with the addition of several data augmentation techniques to the problem of vehicle type recognition was proposed in this research in order to find the best combination of these techniques to solve the problems of the low-quality vehicle image dataset situation.

This paper provides two new vehicle type image datasets, which were collected from Loei Rajabhat University's video surveillance system. The first dataset consisted of 1,310 images, and was named VTID. The second had a total of 4,356 images (VTID2). We then applied ten convolutional neural network architectures, MobileNets (V1 and V2), VGGNet (16 and 19), GoogLeNet (Inception V3, V4, and Inception ResNet V2), ResNet50, and Darknet (Darknet-19 and 53), which had been selected for their fast and accurate properties to challenge the datasets. Finally, the experiments with the combined CNN and the data augmentation techniques were performed.

2. Materials and Methods

2.1 Deep Learning

Deep learning is a successful machine learning technique based on a deep layer neural network. This method has gained popularity in many areas in the fields of artificial intelligence, machine learning, and image processing. A new architecture of deep learning that is more suitable for digital large scale image data has been developed, and it is called a convolutional neural network or CNN. One of the successful CNN pioneers was AlexNet, an eight layers deep convolutional network that first won the ImageNet Large-Scale Visual Recognition Challenge in 2010. After the success of AlexNet, many researchers applied the idea of CNN and developed their model to challenge the predecessor.

Convolutional neural network (CNN) is one of the most advanced algorithms for solving image recognition and detection. This pioneering idea dates back to the 1980s, when it was used to demonstrate how to apply CNN to analyze handwritten digital images [12]. The core structure of CNN is the Convolutional layer, a small filter that creates the feature map from the input image and then combines several features into objects of interest [13]. One famous example of a CNN model was the work of Alex Krizhevsky [7] and his team over the period 2010-2012. They developed a complete CNN model called AlexNet in order to compete in the ImageNet competition. This was followed by VGGNet from Simonyan and Zisserman in 2013 [8], and then Szegedy and a Google team introduced the GoogLeNet or 'Inception' between 2014 and 2016 [9, 14-16]. In 2017, A.G. Howard created a small but efficient network called MobileNets [10].

2.1.1 MobileNet V1

MobileNets is a CNN model intended to reduce the size of the standard CNN model to make it suitable to use in a mobile device. The idea was to replace the general convolutional filters with two separate steps (Depthwise and Pointwise separable convolution), which create a much smaller convolution filter. While Depthwise reduced the size of the length and width dimensions, Pointwise decreased the size of the filter in the depth direction [17]. This network achieved better performance than the other large CNN models and had a smaller network size. These factors made it a suitable option for researchers dealing with the problem of large scale training data.

The Depthwise and Pointwise separable convolutions are ideas for reducing the computation time in the CNN structure [18]. This idea is to split the feature maps from one 3D convolutional layer into two 2D layers, first reducing the depth dimension, and then reducing the height and the width dimensions (see Figure 2). This technique helps decrease the runtime and numbers of parameters, which makes it a small and fast CNN network.

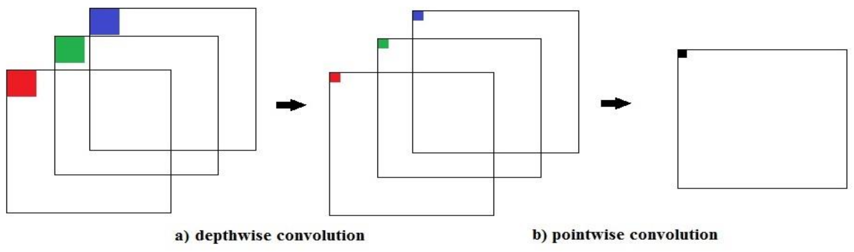

Figure 2. Illustration of the (a) Depthwise and (b) Pointwise separable convolution

Moreover, the designer of MobileNets applied two more hyperparameters that could make an even smaller model possible; width and resolution multipliers. The width multiplier is a parameter used to reduce the number of separable filters that split from the full convolution. For example, if choosing convolution size N x N x M, the standard MobileNets will separate them into M filters with N x N x 1 size. However, the width parameter (α) will replace them with αM filters instead.

The resolution multiplier parameter is the factor that directly reduces the size of the input image from the beginning of the first layer. By reducing the size of the starting image, the total number of computation filters along the way to the final layer is decreased.

The creator of MobileNets designed it to normally use four values for each multiplier. For the width multiplier, researcher can choose four values (1, 0.75, 0.5, and 0.25), and another four for the resolution multiplier (224 x 224, 192 x 192, 160 x 160, and 128 x 128).

2.1.2 MobileNet V2

A new development on MobileNets, MobileNet V2, was introduced in 2018 [19]. The new version had two improved features, Linear Bottleneck and Inverted Residual Block. The Linear Bottleneck layer was used to capture a linear transformation after the ReLU layer if some non-zero values remained (Figure 3). Inverted Residual Block was an idea to improve the standard residual block but had a thinner layer (bottleneck) instead of the thick one (see Figure 4). MobileNet V2 is a bit smaller than the first generation. The number of parameters has been reduced from 4.2 M to 3.4 M but still has an equivalent performance.

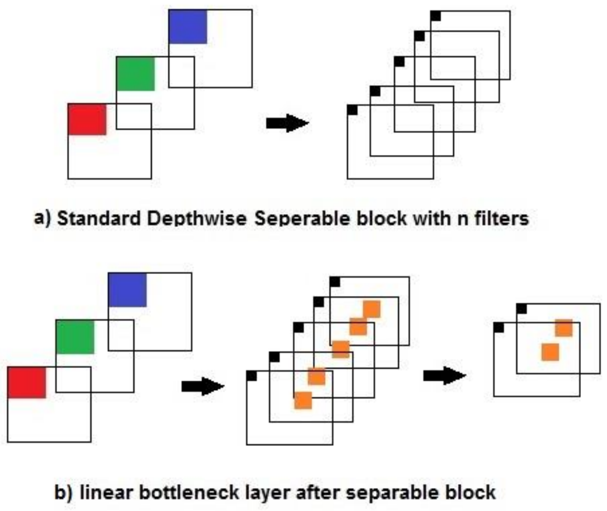
a) Standard Depthwise Seperable block with n filters
b) linear bottleneck layer after separable block

Figure 3. The structure of new separable block with linear bottle neck
a) Standard separable block b) New separable block with linear bottleneck

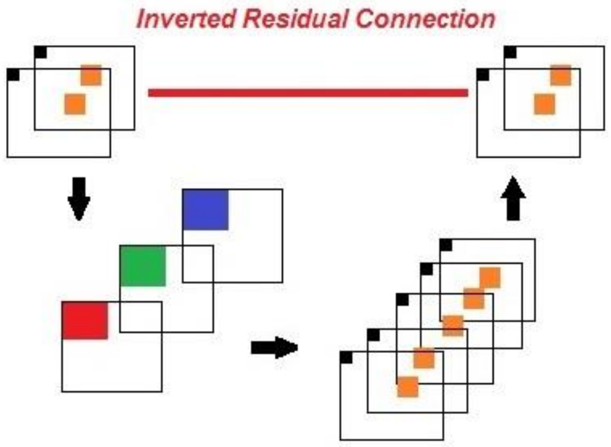

Figure 4. Inverted Residual Block connected to previous bottleneck layer instead of a convolution layer

2.1.3 VGGNet

In 2014, Simonyan and Zisserman proposed the VGG network architecture [8]. This model used only 3×3 convolutional layers on top of the standard CNN network (max pooling, fully-connected, and softmax layer). The results showed that the performance of the VGG network surpassed the other models at that time. The successful VGG architecture came in two sizes, VGG16 & 19. The number 16 and 19 referred to the numbers of weight layers in the network. VGG19 was better than VGG16 because it had a deeper layer but unfortunately this property increased the size of the model.

2.1.4 GoogLeNet

GoogLeNet or Inception V1 is a CNN model created by Christian Szegedy and his team in 2014. In the first version, it had 22 convolution layers and an extra layer called the inception module. Each inception module was a construction of the different sizes for each convolution node (1×1, 3×3, and 5×5) and 3×3 max-pooling node. Inception V1 was the winner at ILSVRC14 and surpassed the performance of VGGNets [9].

After the success of Inception V1, GoogLeNet implemented several updated versions following on from the original. Inception V2 [14] improved the training performance of the networks by using the batch normalization technique. The next model was the Inception V3 [15]. This time they factorized the convolution node in the Inception module and made it smaller. The latest version is Inception V4 [16]. This network is a redesign of Inception V3 with more uniform architecture and more inception modules. Performance evaluations showed that Inception V4 had the smallest error rate on several datasets compared to the previous three networks but required more runtime than Inception V3.

2.1.5 ResNet 50

ResNet, introduced in 2015, was the first CNN that used the concept of residual connection. Its concept was to create a shortcut connection using residual learning block (Figure 5). The aim of these shortcuts was to reduce training error for a deep layer CNN model and it successfully outperformed the VGG network at that time [20].

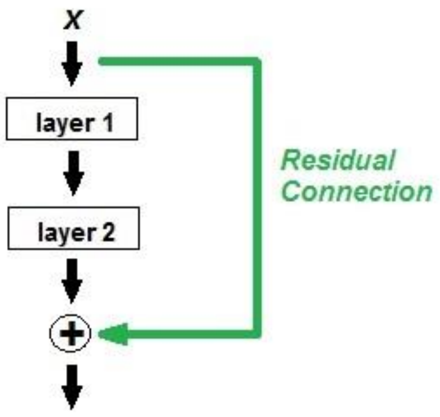

Figure 5. Residual learning block creates a shortcut connection (green arrow) to reduce the path from input to output layer.

2.1.6 Inception ResNet V2

Inception ResNet V2 is the combination concept of inception network and residual connection. It was originally introduced along with inception V4 and Inception ResNet V1 and designed with the aim of using the residual connection to lessen the training process of Inception V3. While V4 was a larger version of Inception V3, Inception ResNet V1 was a combination of V3 and residual connection, and the Inception ResNet V2 was the V4 plus residual block idea. According to Ioffe and Szegedy in 2015 [14], Inception ResNet V2 had the best performance compared to the other three in terms of accuracy. Moreover, it offered improved training speed compared to inception V4.

2.1.7 Darknet-19 and Darknet-53

Another interesting CNN network is called "Darknet". It is the core structure of the object detection technique "YOLO" [21]. YOLO was a “pioneer” project involved in the use of only one neural network to predict both object location and object type in the image. The architecture of YOLO can be separated into two parts: the bounding box detection and the image classification. Originally, YOLO V1 used almost the same structure as GoogLeNet (creating inception module with convolution layer).

YOLO V2 was followed in 2017 with the new classification part called "Darknet-19" [22]. Darknet-19 architecture is mostly similar to VGG19 (19 convolution layer) but it has a special module that seems like an inception block inside the convolution layer. Another model derived from the YOLO series is "Darknet-53". Darknet-53 forms the backbone of YOLO V3 and it is bigger than Darknet-19 (with 53 convolution layers) and includes a residual connection like Inception Resnet. The YOLO V3 was reported to be slightly worse than a previous version but it returned a good result on the detection metric of .5 IOU [23].

2.2 Data augmentation

Image classification with deep learning has been carried out with good results but requires a large amount of training data to avoid overfitting [24]. In some situations, researchers were able to collect enough data to train the model. However, in many situations this was not possible because of a lack of resources. One method to solve this problem is data augmentation, a technique that attempts to create more data samples by adjusting or transforming the original image in various ways.

Data augmentation can be accomplished both with some basic transformations and by using other advanced methods to generate a larger training dataset [25]. Some popular basic augmentations are the affine transformations (Flipping, Rotating, Zooming, and Shifting). These techniques try to create a new image using basic image transformation, for example, by moving the pixel, reflecting the image vertically or horizontally, or by rescaling or cropping the image. Another method involves the use of color modification. Researchers can adjust an image by changing its color system, converting the color image into a black-white image or vice versa, and enhancing the contrast or brightness of the image.

3. Results and Discussion

3.1 Datasets

3.1.1 Vehicle Type Image Dataset (VTID)

The main objective for the use of an image dataset was to examine the five vehicle types of motor vehicles that were the most commonly used ones in Thailand (sedan, hatchback, pick-up, SUV, and van). The recording devices to collect the images were part of a video surveillance system located at Loei Rajabhat University in Loei province, Thailand. The collection process took place during the daytime for four weeks between July and December 2018. Two cameras were installed at the front gate of the university. However, a small number of the van images was produced in the dataset compared to the number of images of the other four vehicle types. Because of this, the researchers decided to add other vehicle-type images such as those of motorcycle into the van group and changed the name of the group to “other vehicles” (Figure 6) to increase diversity. Finally, the first dataset called "Vehicle Type Image Dataset (VTID)" had a total of 1,310 sample images that could be separated into vehicle types as follows; 400 sedans, 478 pick-ups, 129 SUVs, 181 hatchbacks, and 122 other vehicle images. Each image was collected using the 224x224 resolution.

3.1.2 Vehicle Type Image Dataset 2 (VTID2)

After creating VTID, the researchers decided to extend the collection process to create another larger dataset to add further diversity to the dataset in order to avoid data overfitting. Finally, the new dataset, called "Vehicle Type Image Dataset 2 (VTID2)", consisted of 4,356 image samples that could be separated into five vehicle type classes as follows: 1230 sedans, 1240 pick-ups, 680 SUVs, 606 hatchbacks, and 600 other vehicle images.

Moreover, the five data augmentations (Horizontal Flip, Random Shift, Rotation, Zoom, and Brightness) were applied to the VTID2 dataset, which created five larger datasets in the experiment (VTID2-Flip, VTID2-Shift, VTID2-Rotation, VTID2-Zoom, and VTID2-Brightness) (Figure 7). These five datasets were generated using the Keras ImageDataGenerator library with parameter settings for Shift range (-50 to 50), Rotation degree (-45 to 45), Brightness (0.2 to 1.0), Zoom range (0.5 to 1.0). They were first used in the experiment, then later combined (VITD2-All) in the final test.

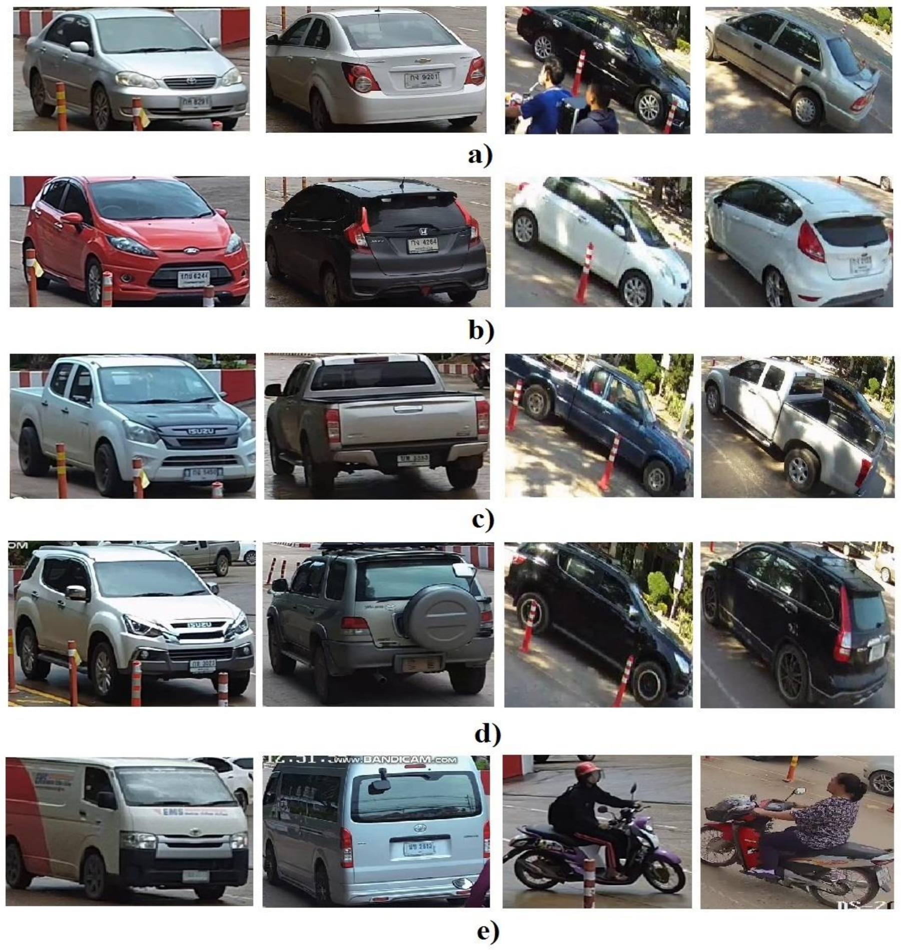

Figure 6. Example of VTID1 dataset collected in four different views (front, back, left, and right): a) Sedan, b) Hatchback, c) Pick-up, d) SUV, e) Other vehicles

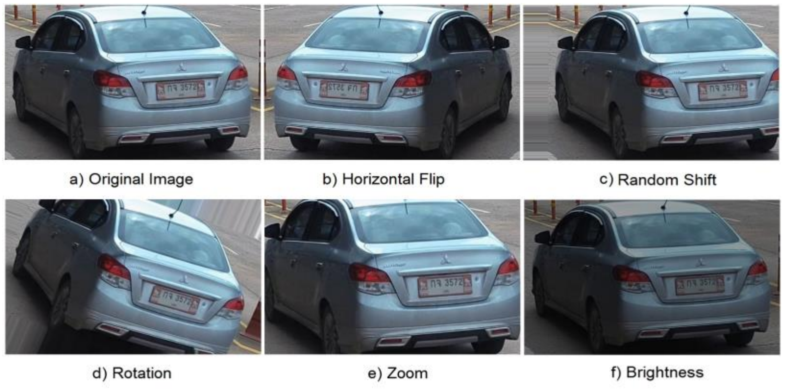

Figure 7. VTID2 with data augmentation (a) original image, (b) horizontal flip, (c) random shift, (d) rotation, (e) zoom, and (f) brightness

3.2 Experiments

3.2.1 Experimental settings and results

For the experiment, the model evaluation was designed based on a 10-fold cross-validation for comparing both the accuracy (Eq.1) and standard deviation of the first VTID dataset. The dataset was separated into training, validation, and test sets, containing 917, 131, and 262 images, respectively. Then, five types of vehicle images; sedan, pick-up, SUV, hatchback, and other vehicles were divided randomly into each set.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{1}$$

where $TP = True\ positive;\ FP = False\ positive;\ TN = True\ negative;\ FN = False\ negative$

Then, several standard CNN architectures were applied to perform image classification experiments on VTID. The overall experiments were run using Python 3.7 with TensorFlow and Keras library on one CPU (Intel Core i-7, 8th Gen, 4.0 GHz, Ram 8 GB). All training processes were trained from scratch for 20 epochs, without using any pre-trained models, to observe the real accuracy and training time of each model.

3.2.2 Experiments with the MobileNets

In the first part of the experiment, the test was run only on MobileNets architecture, but it included different values of its two hyperparameters, the Width and Resolution multipliers. The first results show the performance obtained from MobileNets with various Width Multiplier values on the VTID dataset (Table 1). The result indicated that the number of width multiplier directly affected the performance accuracy and the size of the model. In this case, the size of the model decreased markedly in relaion to the value of the width multiplier (from 16, 10, 5, and 2 MB). For this reason, the accuracy of vehicle type recognition also steadily decreased from 93.40% to 84.33%.

Table 2 shows the results based on another hyperparameter, the Resolution. The results were different from the previous one, but the best value was still the combination of 224x224 resolution and 1.0 width multiplier. However, the size of the resolution value of the input image did not directly affect the size of the model as happened for the width parameter.

Table 1. Performance of MobileNets on different width multipliers (Fixed Resolution = 224)

Width Multipliers	Accuracy (%)	Size (MB)
1.0	93.40	16
0.75	88.10	10
0.5	84.33	5
0.25	86.66	2

Table 2. Performance of MobileNets on different resolution multipliers (Fixed Width = 1.0)

Resolution	Accuracy (%)	Size (MB)
224 x 224	93.40	16
192 x 192	92.54	16
160 x 160	91.20	16
128 x 128	90.39	16

The extension of the Mobinet experiment was performed with every combination of these two hyperparameters. The results are presented in Table 3 which confirmed that the 224x224 resolution and 1.0 width multiplier was the best combination for the MobileNets network for the VTID dataset.

Table 3. Accuracy of MobileNets on every combination of resolution and width multipliers

Resolution	**Width Multipliers**			
	1.0	**0.75**	**0.5**	**0.25**
224 x 224	**93.40**	88.10	84.33	86.66
192 x 192	92.54	87.27	83.21	85.24
160 x 160	91.20	86.66	82.67	84.33
128 x 128	90.39	84.33	82.72	83.54

The analysis of the MobileNets performance is shown in the form of a confusion matrix in Table 4. The highest error type of vehicle was SUV with the lowest accuracy of only 70.54%. This mistake with the SUV probably mainly occurred because it was more similar to the other three groups of vehicle. The SUV had front and side shapes that were close to sedan and pick-up, and most of its shape (front, side, and rear shape) was quite similar to a hatchback.

Table 4. The average confusion matrix of the MobileNets architecture

Actual Class	**Prediction Accuracy (%)**				
	Sedan	**Pick-up**	**SUV**	**Hatchback**	**Other Vehicles**
Sedan	95.25	1	0.75	3	0
Pick-up	0.21	98.33	1.46	0	0
SUV	6.98	11.63	70.54	10.85	0
Hatchback	9.39	1.66	1.11	87.84	0
Other Vehicles	0	0	0	0	100

Examples of SUV errors are shown in Figure 8. Images in the first row are examples of SUV type vehicles that have been classified as Pick-Ups (Figure 8a), then as Sedans (Figure 8b), and finally as Hatchbacks (Figure 8c).

3.2.3 Comparison of the MobileNets and Other CNN Architectures

In the second experiment, the MobileNets architecture was challenged against four other predecessors: CNN (VGG16, VGG19, Inception V3, and Inception V4) on the VTID dataset. The best parameter values from the previous MobileNets experiments (resolution multiplier = 224 × 224 pixels and width multiplier = 1.0) were selected. The results are shown in Table 5. The results show that MobileNets architecture significantly outperformed all of the other four CNN methods.

The results in Table 5 show that MobileNets architecture significantly outperformed four large-scale CNN architectures (Inception V3 and V4 and Vgg16 and 19) in terms of accuracy, image resolution, and size of the model. To summarize the results, the accuracy of MobileNets was 93.40% followed by Inception V4, Inception V3, VGG19, and VGG16 with accuracies of 90.36, 88.81, 79.77, and 77.53%, respectively. The results also show that MobileNets outperformed Inception V4 by more than 3%, and had the lowest uncertainty with a standard deviation rate of only 0.95.

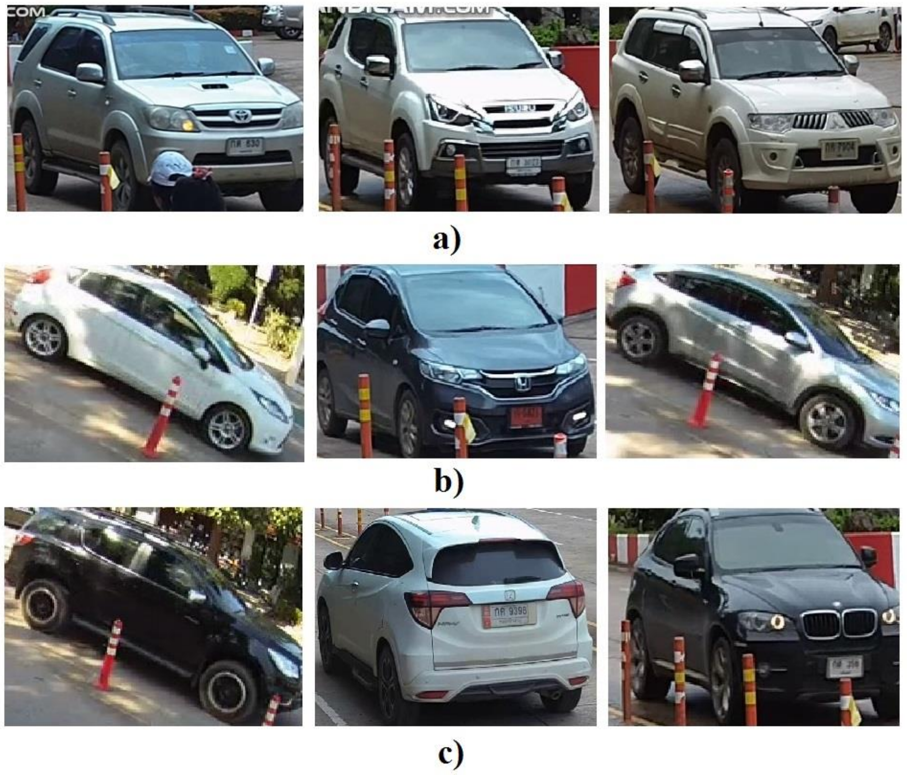

Figure 8. Examples of the wrong result of the SUV when predicting to be (a) Pick-up, (b) Sedan, and (c) Hatchback

Table 5. The average recognition accuracy and the standard deviation of the CNN architectures computed on our vehicle type image dataset

Network Model	Image Resolution (Pixel)	Accuracy ± S.D.	Size (MB)
MobileNets	224	93.40 ± 0.95	16
Inception V4	299	90.36 ± 1.58	163
Inception V3	299	88.81 ± 1.80	83
VGG19	240	79.77 ± 2.04	548
VGG16	240	77.53 ± 2.22	527

In terms of model size, MobileNets was also the best model with the smallest size. The total size of the MobileNets network was only 16 MB, which was ten times smaller than Inception V4 (163 MB). Another interesting issue was that the MobileNets outperformed Inception V4 in accuracy even with a smaller size. Furthermore, Inception V4 needed to be double the size of Inception V3 (83 MB) in order to obtain a higher accuracy than Inception V3.

For the VGG networks (VGG16 and VGG19), VGG19 shows better accuracy than VGG16 but needed a larger network. This was the same issue as with the Inception networks mentioned above. However, the size of both networks was large, 548 MB for VGG19 and 527 MB for VGG16, while the accuracy of VGGNet was the lowest amongst all five networks making it a poor choice for this problem.

The average runtimes for five CNN networks (MobileNets, Inception V4, Inception V3, VGG19, and VGG16) are shown in Table 6. The results showed that MobileNets was the fastest

Table 6. Evaluate of the runtime performances of CNN networks on our dataset

Network Model	Training Time (Min.)	Test Time (Min.)
MobileNets	18.25	7.29
Inception V4	27.53	18.30
Inception V3	22.40	10.21
VGG19	37.21	29.56
VGG16	35.38	27.22

model both in training time and test time, with average runtimes about 18.25 min on training and 7.29 min on the testing process. The second fastest was Inception V3, which had a training runtime of 22.40 min and testing time of 10.21 min. Inception V4 was the third-placed model for speed, with 27.53 min and 18.30 min on training and test. Moreover, VGGNets (16 and 19) were still the poorest networks with the slowest runtimes compared to the other three networks, with both training and test processing times around 30 min.

From the result shown in Tables 5 and 6, MobileNets outperformed the other four networks for vehicle type recognition in every aspect (accuracy, size, and speed).

3.3 Experiments with Data augmentation

In the third and final experiment, the performances of MobileNets and other CNN models were tested with two main proposes. First, the higher diversity of the vehicle type image dataset with larger data examples (VTID2) and additional data augmentation techniques. Second, the newer CNN models with concepts of Residual Block Connection.

For this section, experiments were run on seven image datasets that were created after adding the data augmentation techniques mentioned in section 3.1.2 (VTID2, VTID2-Flip, VTID2-Shift, VTID2-Rotation, VTID2-Zoom, VTID2-Brightness, and VTID2-All) and using 8 convolutional neural network architectures: MobileNets, Inception V3, Inception V4, ResNet50, Inception ResNet V2, Darknet-19, Darknet-53, and MobileNetV2 (excluding VGG16 and VGG19) with 10-fold cross-validation as in the previous experiment. The standard VTID2 had 4,356 images, while the other data augmentation dataset was double the size of the first one. Finally, the VTID2-All combined all images from every data augmentation method to make the total number of images of 26,136 (6 times of VTID2).

The results of data augmentation showed some surprises in the performance of the first MobileNet (see Table 7). In terms of total accuracy, MobileNets outperformed the other seven networks in every dataset. MobileNetV2 and Darknet-53 were close competitors, however MobileNetV2 scored 5 of 7 augmentation types. Inception Resnet V2 proved superior to both Inception V3 & V4 and ResNet50 but finished behind the MobileNets and Darknet series.

For the effects of data augmentation, the best augmentation-technique was the brightness method. It had the highest score in 5 of 7 models excluding only Inception V3 & V4. MobileNets had the highest accuracy (95.46%) with data brightness. The worst method in our experiment was Image Rotation. It had the lowest score but also surprisingly the same ratio (5 of 7).

Unfortunately, combining every data augmentation (VTID2-All) did not produce significant improvement in the VTID2 dataset. The accuracy of VTID2-All was around the lowest score in every CNN architecture compared to using only one augmentation.

Table 7. The average recognition accuracy (%) using data augmentation

Network Model / Dataset	VTID2	VTID2 Flip	VTID2 Shift	VTID2 Rotation	VTID2 Zoom	VTID2 Brightness	VTID2 All
MobileNets	**94.38**	**95.38**	**94.98**	**93.93**	**94.19**	**95.46**	**93.98**
Inception V4	92.17	90.77	90.02	87.28	88.07	91.14	86.40
Inception V3	91.37	91.45	89.49	88.06	86.83	91.01	87.95
ResNet50	91.24	91.33	89.57	87.85	87.02	90.95	87.65
Inception ResNet V2	92.95	93.71	94.27	89.86	90.01	94.65	89.93
Darknet-19	91.58	91.98	91.32	88.65	87.04	92.07	89.13
Darknet-53	93.99	94.54	94.02	91.87	92.01	94.98	92.87
MobileNetV2	93.56	94.32	94.65	92.85	92.89	95.22	93.60

For the training runtime (see Table 8), the MobileNets family was still the fastest model. At that time, MobileNetV2 was the most active network because of its smaller size compared to its parent. Inception ResNet V2 was a little slower than Inception V4 but had higher accuracy. ResNet50 was slightly slower than Inception V3 but was close in terms of accuracy. The speed of Darknet-19 was nearly the same as that of MobileNets but its accuracy was different.

Table 8. Evaluation of the training time performance of CNN networks after data augmentation

Network Model	VTID2 (min.)	VTID2-ALL (min.)
MobileNets	50.72	259.24
Inception V4	81.54	453.63
Inception V3	65.87	359.48
ResNet50	74.21	402.56
Inception ResNet V2	91.14	502.25
Darknet-19	53.22	284.61
Darknet-53	102.32	547.93
MobileNetV2	**42.18**	**215.26**

Finally, the new confusion matrix for the VTID2 dataset and Brightness augmentation (Table 9) showed the impressive improvements of each vehicle classification rate, and especially on SUV and Hatchback. The accuracy of SUV rose from 70.54% to 88.23%, and for Hatchback it increased from 87.84% to 92.07%. This confirmed that increasing the size of the image dataset and using data augmentations techniques could help to improve the overall performance of image classification.

Table 9. The new confusion matrix of VTID2 with MobileNets architecture and Brightness augmentation.

Actual Class	Prediction Accuracy (%)				
	Sedan	Pick-up	SUV	Hatchback	Others Vehicle
Sedan	**96.75**	0.8	0.5	1.95	0
Pick-up	0.18	**98.39**	1.43	0	0
SUV	2.95	4.85	**88.23**	3.97	0
Hatchback	5.81	1.47	0.65	**92.07**	0
Others Vehicle	0	0	0	0	**100**

3.4 Discussion

The results from the previous section show the outstanding performance of MobileNets in these vehicle type recognition experiments. This model demonstrated the highest accuracy in both datasets (VTID and VTID2) and also required the least training time to create a model. However, to compare the real accuracy and runtime of each CNN model, the researchers decided to run the experiment by training each model from scratch without bias from the pre-trained parameters and trained for only 20 epochs to compare speed. These might be the issues for further study if the pre-trained values or longer training processes affect the results.

4. Conclusions

In this paper, the experiments concerned with the use of deep learning models to solve the issues of vehicle type image recognition were performed. Ten convolutional neural networks (CNNs) were chosen to be used in this paper (MobileNets, VGG16&19, Inception V3, Inception V4, ResNet50, Inception ResNet V2, Darknet-19, Darknet-53, and MobileNetV2) to compare the performance of each architecture. The results indicated that MobileNets was the best method to deal with vehicle type recognition problems, in terms of speed, accuracy, and size. The accuracy of MobileNets was the best, its model size was the smallest, and its runtime was the fastest compared to the other architectures. The fact that MobileNets is the best, fastest, and smallest size makes it suitable to be used in every computing platform, including mobile devices which have slower speeds and smaller memories. Additionally, the SUV vehicle type proved to be the worst model to classify because it was similar in shape to other vehicle types, including Pick-up and Hatchback. Incidentally, this might have been affected by an unbalanced amount of each image class in the training data, and will require further work to create a new unbiased dataset with a larger size and the same amount of images in every vehicle type.

Another interesting result was Mobilenets' parameter-tuning. It was found that both width multiplier and resolution multiplier did not help to solve this problem. These two parameters were able to reduce the MobileNets size, but did not enhance accuracy. Moreover, with data augmentation, the results suggested that MobileNets still won against other CNNs in every single augmentation technique. The best augmentation method found in our research was Image Brightness. The combination of our second dataset (VTID2), brightness augmentation, and MobileNets V1 produced the highest accuracy (95.46%). For future work, it will be interesting to add some improvements inside the architecture of MobileNets (or other models) to increase accuracy rate and reduce runtime when dealing with larger datasets.

5. Acknowledgements

The data collection processes in this research were supported by the Office of Academic Resources and Information Technology at Loei Rajabhat University, Thailand.

References

[1] Li, J., Zhao, W. and Guo, H., 2009. Vehicle type recognition based on Harris Corner Detector. *Proceedings of the Second International Conference on Transportation Engineering*, Chengdu, China, July 25-27, 2009, pp. 3320-3325.

[2] Zhang, B., 2013. Reliable classification of vehicle types based on cascade classifier ensembles. *IEEE Transaction on Intelligent Transportation Systems*, 14, 322-332.

[3] Clady, X., Negri, P., Milgram, M. and Poulenard, R., 2008. Multi-class vehicle type recognition system. *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, Springer, Berlin, 228-239.

[4] Dong, Z., Wu, Y., Pei, M. and Jia, Y., 2015. Vehicle type classification using a semisupervised convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 16(4), 2247-2256.

[5] Huttunen, H., Yancheshmeh, F.S. and Chen, K., 2016. Car type recognition with deep neural networks. *Proceedings of the Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 2016, pp. 1115-1120.

[6] Bautista, C.M., Dy, C.A., Mañalac, M.I., Orbe, R.A. and Cordel, M., 2016. Convolutional neural network for vehicle detection in low resolution traffic videos. *Proceedings of the 2016 IEEE Region 10 Symposium (TENSYMP),* Bali, Indonesia, 2016, pp. 277-281.

[7] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems,* December, 2012, pp. 1097-1105.

[8] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. [online] Available at: http://arxiv.org/abs/1409.1556.

[9] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 1-9.

[10] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. [online] Available at: http://arxiv.org/abs/1704.04861.

[11] Boonsirisumpun, N. and Puarungroj, W., 2018. Loei Fabric Weaving Pattern Recognition Using Deep Neural Network. *Proceedings of the 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Nakhonpathom, Thailand, July 11-13, 2018, pp. 1-9.

[12] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computing*. 1(4), 541-551.

[13] LeCun, Y. and Bengio, Y., 1995. Convolutional networks for images, speech, and time series. In: M.A. Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge: MIT Press, pp. 1-14.

[14] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of Machine Learning Research,* 2015, pp. 448-456.

[15] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2818-2826.

[16] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A., 2017. Inception-v4, Inception-Resnet and the impact of residual connections on learning. *Proceedings of the Thirty-First AAAI (Association for the Advancement of Artificial Intelligence) Conference on Artificial Intelligence*, San Francisco, CA, USA, 2017, pp. 4278-4284.

[17] Yoo, B., Choi, Y. and Choi, H., 2018. Fast Depthwise Separable Convolution for Embedded Systems. *International Conference on Neural Information Processing*, Siem Reap, Cambodia, December 13-16, 2018, 656-665.

[18] Sifre, L. and Mallat, S., 2014. *Rigid-Motion Scattering for Image Classification.* Ph.D. ENS University, Paris, France.

[19] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510-4520.

[20] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.

[21] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 779-788.

[22] Redmon, J. and Farhadi, A., 2017. YOLO9000: Better, faster, stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263-7271.

[23] Redmon, J. and Farhadi, A., 2018. *Yolov3: An Incremental Improvement*. [online] Available at: https://arxiv.org/pdf/1804.02767.pdf.

[24] Shorten, C. and Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 60, https://doi.org/10.1186/s40537-019-0197-0.

[25] Mikołajczyk, A. and Grochowski, M., 2018. Data augmentation for improving deep learning in image classification problem. *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 117-122.