Hyper-parameters optimisation of deep CNN architecture for vehicle logo recognition

Foo Chong Soon\textsuperscript{1}, Hui Ying Khaw\textsuperscript{1}, Joon Huang Chuah\textsuperscript{1} \& Jeevan Kanesan\textsuperscript{1}

\textsuperscript{1}Department of Electrical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia
\textsuperscript{2}E-mail: jhchuah@um.edu.my

Abstract: The training of deep convolutional neural network (CNN) for classification purposes is critically dependent on the expertise of hyper-parameters tuning. This study aims to minimise the user variability in training CNN by automatically searching and optimising the CNN architecture, particularly in the field of vehicle logo recognition system. For this purpose, the architecture and hyper-parameters of CNN were selected according to the implementation of the stochastic method of particle swarm optimisation on the training–testing data. After obtaining the optimised hyper-parameters, the CNN is fine-tuned and trained to ensure better network convergence and classification performance. In this study, a total of 14,950 vehicle logo images are divided into two independent training and testing sets. In addition, these images are segmented coarsely, thus the requirement of precise logo segmentation is obviated in this work. The learned features of the CNN were sufficiently discriminative to be classified using multiclass Softmax classifier. With implementation using a graphics processing unit (GPU), the computation time of the proposed method is acceptable for real-time application. The experimental results explicitly prove that the authors’ approach outperforms most of the state-of-the-art methods, achieving an accuracy of 99.1\% over 13 vehicle manufacturers.

1 Introduction

In recent years, many cutting-edge technologies have been implemented in the domain of intelligent transportation systems (ITSs), particularly to improve mobility and public transportation safety. Vehicle manufacturer recognition (VMR) and vehicle licence plate recognition are two important areas in the field of ITS. Vehicle information such as number plate and vehicle make (manufacturer) can be equally crucial in various fields especially border crossing control. Generally, the number plate information of foreign vehicles is not recorded in the vehicle licencing databases of a government agency. Thus, the vehicle maker and other information are not possible to be retrieved using its number plate. In such a circumstance, the recognition of vehicle logo appears to be more useful in providing the vehicle maker information. Apart from that, in cases when counterfeit vehicle licence plates are illegally used on stolen vehicles, the vehicle logo recognition (VLR) is an important function. It can be implemented by the governmental authority in tracking the mismatch between the vehicle maker and its plate number from the system database, thus confirming counterfeit cases. Furthermore, these vehicle recognition systems are also crucial in providing car ownership statistical and analytical information to business entities as well as governmental bodies [1].

Vehicle licence plate is a discriminative feature that is widely used to recognise each individual vehicle. In contrast with vehicle licence plate location (LPL), which has already been widely studied and addressed for dozens years ago, VLR is still a demand in the research area of ITS. In other words, VLR is challenging due to its necessities for precise vehicle logo localisation and segmentation, robustness against various imaging conditions, as well as real-time application.

Most of the common logo-based VMR systems are generally dual-stage process, which consists of precise vehicle logo detection (VLD) and VLR. Canny edge detector has been proposed by Munroe and Madden [2] for feature extraction of vehicle frontal view images. They achieved an overall accuracy of 97.46\% by deploying the k-nearest neighbour classification algorithm. In spite of that, there is still a difficult task for the system to cope with a wide range of different vehicles’ physical appearances, i.e. colours and shapes. Gradient filters are utilised by Petrovic and Cootes [3] to locate several region of interests (ROIs) such as vehicle licence plate, vehicle logo, and grille part. They implemented various feature extraction approaches, i.e. square mapped gradients, Sobel mapped gradients etc., for recognition of vehicle manufacturer (make) and model. However, their proposed method is weakened in a wider range of viewpoints and also in plane rotations. A novel approach using Harris corner strengths has been proposed by Pearce and Pears [4] for feature detection in the vehicle make and model recognition. By adopting a Naïve Bayes classifier, they managed to further improve their classification rate from 94.9 to 96\%. Psyllos \textit{et al.} [5] proposed a phase congruency method for vehicle mask segmentation. They implemented a probabilistic neural network (NN) for the recognition of vehicle manufacturer logos and also its respective colour. However, their system is still constrained in complex situations such as a wider range of imaging conditions and affine transformations. A lot of approaches have been proposed for VLR; these methods have substantially utilised the effectiveness of low-level features [6, 7], i.e. scale invariant feature transform (SIFT) and histogram of gradients (HOG) to recognise the vehicle makes. Despite outperforming in VLR performance, most of these aforementioned approaches are still heavily dependent on both low-level features and handcrafted features (e.g. SIFT). In other words, these traditional approaches are still constrained in handling images with various imaging conditions, i.e. logo rotation or translation, poor illumination, viewpoints variation, and degradation by noise. Moreover, the precise VLD is crucial in order to ensure the good performance of VLR, if handcrafted features are employed. On contrary, the imprecise detection and segmentation of the logo will subsequently deteriorate the classification performance.

Convolutional NN (CNN) is a hierarchical NN which incorporates numerous convolutional layers alternating between subsampling layers, by mimicking both simple and complex neurons in the primary visual cortex of the brain [8]. In contrast to handcrafted features learned in the aforementioned approaches, CNN is able to learn and extract hierarchical features by constructing low-level features to high-level features through the adaption of its multilayer feed-forward structure. More importantly, CNN is robust against affine transformation and image distortion.
In the recent decade, CNN has been extensively applied in various areas, i.e. image noise-type recognition [9], vehicle LPR [10], vehicle-type classification [11] etc. In the domain of ITS, Huang et al. [1] proposed a CNN which is pretrained with principal component analysis (PCA) to reduce the computational time of training procedure for VLR. However, the unsupervised learning algorithm of pretraining strategy for each layer of the CNN is inconvenient and not sufficiently discriminative [12].

Deep CNN is a system with a combination of neurons structured in the input, hidden, and output layers. The neurons are interconnected to each other by a set of convolutional kernels. Similar with most learning approaches, the performance of a trained CNN model depends greatly on a set of hyper-parameters [13]. In general, hyper-parameters optimisation is implemented to identify an optimal set of hyper-parameters in order to minimise the generalisation error [14]. Many previous research works have been proposed on conducting the trials to select hyper-parameters settings, i.e. grid searches, random searches, sequential, and tree-based searches [15]. Owing to less complicated implementation, effective computation power, and high convergence rate in dealing with an optimisation problem, particle swarm optimisation (PSO) has been adopted in the domain of NN design problem [16].

The concept of swarm intelligence (SI) was introduced in the domain of computing and artificial intelligence, inspired by the collective behaviour of social animals [17]. PSO is an iterative stochastic approach based on SI, inspired by the information sharing and social behaviour observed in fish schools and bird flocks. In addition, PSO is a global optimisation technique for searching the optimal solution in a multidimensional space of the parameters. The social interaction of PSO is direct, as the movement or update of each particle is not only affected by its own best solution but it is also guided toward the best position of the whole swarm [18]. PSO algorithm has been extensively applied in many domains of engineering and computer science such as for vehicle routing [19], multifrequency ground penetrating radar [20], fruit ripeness classification [21], carpool service problem [22], financial time series volatility forecasting [23], and vehicle design process [24].

Inspired by the outstanding performance of CNN and PSO in recognition and optimisation, respectively, in this paper, we propose a CNN architecture which is hybridised and adapted with a simple yet efficient PSO hyper-parameter optimisation method for the VLR system. Rather than substantially depending on ad hoc tricks and painstakingly CNN parameters tuning for training process, an optimal set of hyper-parameters of CNN has sought automatically and pinpointed using PSO. It is crucial to point out that the overall training time is reduced since the process of hyper-parameter tuning for network training is no longer required. In addition, we demonstrate the feasibility of our method in VLR, whereby the computational cost of CNN is greatly reduced after adopting the PSO-optimised hyper-parameters. The trained CNN is a better option of the real-time application when using graphic processing unit (GPU)-accelerated computation.

The next part of this paper is organised as follows. In Section 2, we describe the framework of our proposed CNN for VLR. The experimental results and analysis are presented in Section 3. Eventually, we conclude this paper in Section 4.

2 Framework of the proposed method

In this work, we designed an end-to-end VLR model based on deep NN architecture. The VLR is structured in three main stages: first coarse segmentation of ROI, i.e. vehicle logo in this work, second PSO for CNN hyper-parameter optimisation stage, and finally fine-tuning of CNN for recognition stage.

2.1 Coarse segmentation of ROI

To detect vehicle logos from the PLUS vehicle image dataset, the coarse segmentation technique is implemented and presented in Fig. 1. The original vehicle frontal view images are acquired from the traffic monitoring system as shown in Fig. 1a. Rather than precisely locating the vehicle logo, as indicated by the yellow box in Fig. 1b, the proposed system only needs to coarsely extract a larger area of the vehicle logo as defined by the green box in Fig. 1c, for further processing. Thus, it promises a very high probability that the vehicle logo will be detected in the coarsely segmented region.

These original images are initially converted to a grey-scale format with 8 bit resolution and then a licence plate location (LPL) module is implemented. The initial step of LPL is to perform binarisation on the vehicle images. It is worth noting that the alphabets and numbers on vehicle licence plate are both white coloured in Malaysia. In the connected component labelling process, all connected white pixels are labelled separately as an individual component. Subsequently, the amount of white pixels in each connected component is calculated. Generally, the alphabets and numbers on the car plate have a certain range of white pixels. Using this characteristic, the next stage is to filter out the white pixel regions that do not fall within the preset lower and upper limits. Since the images were captured from a fixed angle at the toll collection booth, the licence plate is located on the bottom part of vehicle image. To reduce the computational time, the sliding concentric window (SCW) segmentation approach is applied across the bottom part of the binarised vehicle image from left to right. At each stride of SCW, the total area of the white pixel region is recorded. The recorded region that captures the maximum area indicates the licence plate, which will be the output of LPL module.

Vehicle logo with size \(L \times L\) is to be retrieved for accommodating various vehicle manufacturer logos in this work. The logo candidates are obtained relative to the detected licence plates. After LPL, the vehicle logo's bottom centre is indicated by the yellow dot and is coordinated as \((x_1, y_1)\) as shown in Fig. 2. The \(L/2\) length of vehicle logo is extended to both left and right from the yellow dot. Then, the \(L \times L\) vehicle logo can be located relatively. To augment and construct the XMUPlus dataset, we segment and include another three additional vehicle manufacturer logos, i.e. Proton, Perodua, and Nissan into the XMU dataset. In this work, we exclude vehicles with the larger size, i.e. suburban utility vehicle and four-wheel drive. In addition, this work is based on the assumption that the coarse segmentation for LPL has 100% accuracy.

2.2 Convolutional NN

The proposed CNN incorporates convolutional layers (Conv), pooling layers (Pool), and a fully connected (FC) layer. CNN functions based on three architectural ideas: (i) shared weights; (ii) local receptive fields; and (iii) spatial or temporal subsampling, in order to provide invariance in translation, rotation, and scaling. The network is fed with almost raw input (e.g. image) that is size normalised and centred. Then, each neuron in intermediate layer receives inputs from a set of units situated in a small neighbourhood of the prior layer. Initially, these neurons extract fundamental visual features such as endpoints, oriented edges, and corners by utilising local receptive fields. These extracted
fundamental features are then concatenated by the subsequent deeper intermediate layers so as to extract higher-level features. Eventually, an output of prediction is resulted based on discriminative features extracted via alternating layers of the network.

2.2.1 Convolutional layers: Convolution layers (Conv1, Conv2, and Conv3) are implemented to convolve the input image or feature maps of the previous layer with different filter masks (or weights) in order to extract features. It comprises two-dimensional weights that are maps. We denote $x_{m}^{\ell-1}$ as the mth input feature map of $\ell-1$ layer, $W_{mn}^\ell$ as the weight filter connecting the nth feature map of the output layer to the mth feature map of the input layer and $b_{n}^\ell$ convolved with the input signal to generate the output feature as a bias. Therefore, we have the output feature map in layer formulated as

$$x_{n}^\ell = f\left(\sum_{m} x_{m}^{\ell-1} * W_{mn}^\ell + b_{n}^\ell\right)$$

where * represents the convolutional operation and f is a rectified linear unit (ReLU) activation function as computed by $f(x) = \max(0,x)$. As compared to the traditional and common activation functions such as hyperbolic tangent (tanh) or sigmoid, the ReLU activation function provides the non-linearity in the feature extraction process which is more effective for training purpose.

2.2.2 Pooling layers: Pooling process is deployed to downsize feature maps and hence provide invariance of the output to shifts and distortions. In addition, this stage will also mitigate the number of the parameters that need to be trained and also avoid overfitting problem. Each feature map in Pool1 and Pool2 is obtained by a subsampling process called max pooling that is implemented on the corresponding feature map in the previous layer as

$$x_{n}^\ell = \text{down}(x_{n}^{\ell-1})$$

where down(⋅) is a type of pooling process. A non-overlapping rectangular region of dimension $(d_{x}, d_{y})$ is implemented for detecting and choosing maximum response over input feature maps of the previous layer, where the downsampling factor $d_{x} = d_{y} = 2$ and stride = 1. Therefore, the size of the output feature map in the current layer will become half of the size of the input feature map in the previous layer. Eventually, multiscale features of the logo images will be extracted at the end of this process. The pooling stages in layer Pool2 will repeat the same procedure as in Pool1. The convolutional layers and the max-pooling layers are repeating alternately to construct a multilayer structure for feature extraction.

2.2.3 FC layer: FC layer acts as nested linear classifiers. In this work, Softmax classifier (or multinomial logistic classifier) is used in the final layer of the network to handle multiple classes of vehicle logo. The feature vector with fixed dimensions, which is the output of the previous layers, will be fed as input to the Softmax classifier. The output of the Softmax classifier computes the probabilities of the vehicle logo classes for each image, in which the one with the highest value represents the predicted vehicle logo class. For a binary classification using logistic regression, the hypothesis is denoted as

$$P(y = 1|x; W) = \frac{1}{1 + \exp(-W^Tx)}$$

where y indicates the class label, $x \in \mathbb{R}^{(K+1)\times 1}$ represents the K-dimensional feature vector, $W \in \mathbb{R}^{(K+1)\times N}$ is the weight vector parameter. For a multiclass vehicle logo classification problem where the response variable y can take on different N values, hence the binary classification can be generalised to generate output probability as follows:

$$P(y = r|x; W) = \frac{\exp(w_{r}^{T}x)}{\sum_{i=1}^{N} \exp(w_{i}^{T}x)}$$

2.2.4 Cross-entropy loss function: Eventually, the generalisation error is calculated at the final Softmax layer. The loss function that is implemented for the Softmax classifier as described in (4) is defined by

$$E(W) = -\sum_{r=1}^{N} \sum_{i=1}^{N} 1\{y^{(i)} = r\} \log \frac{\exp(w_{r}^{T}x^{(i)})}{\sum_{r=1}^{N} \exp(w_{r}^{T}x^{(i)})}$$

where $1\{\cdot\}$ denotes the indicator function, so that 1{a true vehicle logo class} = 1 or 1{a false vehicle logo class} = 0.

2.3 Hyper-parameter optimisation by PSO

Optimisation is the process of finding the best element among a set of possible solutions or alternatives. Mathematically, the optimisation process is accomplished by defining the aim in terms of an objective function f which consists of D parameters that is given by

$$f : \mathbb{R}^{D} \rightarrow \mathbb{R}$$

The value of f, which quantifies the degree of optimality of a particular set of parameters for the specific task, is often referred to...
as the fitness. Our goal is to minimise \( f \) itself in order to find the best values of CNN hyper-parameter \( x_{\text{opt}} \)

\[
x_{\text{opt}} f(x_{\text{opt}}) \leq f(x) \forall x \in \mathbb{R}^D
\]

(7)

where the \( D \)-dimensional space of the function \( \mathbb{R}^D \) represents the search space and the vector of \( x \) is the candidate solutions. The candidate solution \( x_{\text{opt}} \) is the optimal one which minimises \( f \). In PSO, each candidate solution (or particle) denotes the certain point in the \( D \)-dimensional space, where \( D \) is the number of CNN hyper-parameters to be optimised. The position of the \( i \)th particle is defined by the vector \( x_i \) as follows:

\[
x_i = [x_{i_0}, x_{i_1}, x_{i_2}, \ldots, x_{i_D}]
\]

(8)

Subsequently, the swarm constitutes the population with \( N \) candidate solutions

\[
X = \{x_1, x_2, \ldots, x_N\}
\]

(9)

In each PSO generation (or iteration), the position of each particle \( x_i \) is updated corresponds to its newly updated velocity, \( v_i(t+1) \), as formulated

\[
x_i(t+1) = x_i(t) + v_i(t+1)
\]

(10)

With the goal of preventing particles travel outside the boundaries of the search space, the particles’ position is constrained as follows:

\[
v_i(t+1) = \omega(t+1)v(t) + c_1 R_1(p_i - x(t)) + c_2 R_2(g - x(t))
\]

(11)

where \( \omega, c_1 \) and \( c_2 \) are acceleration constants. \( R_1 \) and \( R_2 \) are random numbers generated from a uniform distribution in \([0, 1]\). To overcome the velocity explosion problem, velocity clamping is adopted in order to prevent the particle from exceeding beyond the boundaries of the search space. Thus, a velocity threshold is introduced as follows:

\[
\text{if } v_j(t+1) > v_{\text{max}} \text{ then } v_j(t+1) = v_{\text{max}} \\
\text{if } v_j(t+1) < -v_{\text{max}} \text{ then } v_j(t+1) = -v_{\text{max}}
\]

(12)

3 Experiment

3.1 Dataset descriptions

In this research work, we included another three classes of vehicle logo and extended the VLR dataset from XMU [1] into a larger size of the dataset which is known as an XMUPlus dataset in this work. The XMU dataset is originally consisting of ten classes of vehicle logo, namely Buick, Chery, Citroen, Honda, Hyundai, Lexus, Mazda, Peugeot, Toyota, and Volkswagen. After extension of XMU dataset, the XMUPlus dataset comprises 13 different manufacturer classes which are made up of every logo classes of XMU dataset with additional three logo classes, namely Proton, Nissan, and Perodua as illustrated in Fig. 3. These additional outdoor vehicle frontal images were captured from traffic surveillance camera from the Malaysian North–South Expressway operated by PLUS. All of these vehicle logo images were coarsely segmented and normalised to spatial size 70 \( \times \) 70 px\(^2\) in a grey-scale format. An amount of 13,000 training images and 1950 testing images were utilised for validating the classification performance. For each manufacturer or class, 1000 images were training images and 150 images were testing samples. These vehicle logo images encompass various outdoor imaging conditions, i.e. the large variation of distortions, illumination variance and translation.

3.2 Training of VLR

3.2.1 Particle SO: The proposed methodology and algorithm of CNN hyper-parameter optimisation by PSO is depicted in Fig. 4 and Algorithm 1 (Fig. 5). Several hyper-parameters were sought and optimised over the dataset using PSO in terms of the following hyper-parameters such as learning rate (\( \alpha \)), spatial size of convolutional kernels (\( S_{\text{conv}} \)), and a number of convolutional kernels (\( N_{\text{conv}} \)), in each convolutional layer \( i \).

The search space boundaries for the hyper-parameters are set within a wide range as shown Table 1. To speed up the convergence, each PSO iteration was extended with a narrower search space domain for the learning rate hyper-parameter. After completion of each PSO iteration, mean and standard deviations of each learning rate hyper-parameter among the best 20% solutions (or particle positions) were computed. The upper boundary of hyper-parameter search space was then updated according to the mean (\( \mu_i \)) and standard deviation (\( \sigma_i \)) as follows:

\[
UB_i = \mu_i + \sigma_i
\]

(15)

Although \( \mu_i + n\sigma_i \), where \( n = 2, 3, \ldots \), would generate a more robust search space, we implement a narrower range to increase the rate of convergence. The search space of only categorical hyper-parameter, i.e. learning rate, was considered and narrowed from the optimisation process at the end of each PSO iteration, as the
The learning rate will greatly affect convergence. To ensure a good convergence outcome, the upper boundary of the learning rate search space was preferably adjusted iteratively while ensuring its range is neither extremely high nor too low.

### 3.2.2 Convolutional NN

In our adaptation domain, we fine-tune our CNN model using the optimised hyper-parameters and structures sought by PSO. The MatConvNet library developed by Vedaldi and Lenc [25] is used for all the experiments involving CNN. Hierarchical features of vehicle logo images are extracted by deploying CNN trained by BP and SGD. A ten-way output layer (after the FC layer) has been implemented along with weights that are randomly initialised from a normal distribution. SGD is implemented to update each weight parameter in every iteration. SGD is scalable to very large datasets with satisfactory convergence capability in deep learning [26]. The gradient of the parameters is computed using only a minibatch of training examples. The new update in every iteration is formulated as

$$W_{i+1} = W_i - \alpha \nabla_W E(W)$$

where $\alpha$ denotes the learning rate and the parameter update is approximated by computing the cost and gradient with respect to a minibatch size of the training set. In our experiment, we use $\alpha = 4 \times 10^{-2}$ and minibatch size $= 256$. The utilisation of minibatch for parameter update in SGD minimises the variance of parameter update and thus achieves more stable convergence during the training process. All the experiments are conducted using MATLAB (R2015b) environment running on a desktop with an Intel(R) Core(TM) i5–6400 central processing unit 2.7 GHz and an Nvidia GeForce GT 730 GPU.

### 3.3 Result and discussion

A three-stage system has been proposed for VLR in this research. In the first stage, we locate and then segment the vehicle logos from images. For the second stage, we introduce a hyper-parameters optimisation by using PSO in order to search for a minibatch size of the training set. In our experiment, we use $\alpha = 4 \times 10^{-2}$ and minibatch size $= 256$. The utilisation of minibatch for parameter update in SGD minimises the variance of parameter update and thus achieves more stable convergence during the training process. All the experiments are conducted using MATLAB (R2015b) environment running on a desktop with an Intel(R) Core(TM) i5–6400 central processing unit 2.7 GHz and an Nvidia GeForce GT 730 GPU.
3.3.1 CNN hyper-parameters optimisation by PSO: The maximum accuracy is achieved by various CNN architectures via multiple PSO experiments as shown in Table 2. We expect the loss function of CNN drops within the selected CNN training epochs by using a set of hyper-parameters generated by PSO. In other words, each PSO run will evaluate the fitness of particles solution based on the cost value after five training epochs of CNN. The selection of the number of CNN training epoch is crucial as: if the training epoch of CNN is too low, the tendency of the CNN to converge is low or uncertain. On the other hand, the PSO training process takes time exponentially if the training epoch of CNN is too high. For each PSO run, a constant number of five training epochs is utilised for calculating the fitness of particles solution based on the objective function of CNN. We have evaluated various numbers of training epochs and eventually selected five epochs which are ample to reduce computation time without deteriorating the convergence of network.

We have varied the number of PSO iteration corresponding with the number of PSO population in order to evaluate the effect of the number of particle population on the swarm evolution and on the classification performance of various CNN architectures, which are sought by PSO. To this purpose, the results obtained with a population of ten particles were compared with respect to those swarms of 15 or 20 individuals. As either the iteration number or population number increases, the training time increases subsequently. In general, it was observed that as either the number of iteration or population size increases, every maximum accuracy achieved by various CNN architectures in XMU dataset and XMUPlus dataset are still staying above 99 and 98%, respectively. Thus, a CNN architecture with optimised hyper-parameters is sobering to account for its optimum classification performance in conjunction with satisfying training time. Owing to the shortest PSO training time and high classification accuracy, the CNN architectures of (1a) and (1b) are chosen for XMU dataset and XMUPlus dataset experiments, respectively.

3.3.2 CNN fine-tuning for VLR: We have adapted the CNN architecture searched by PSO, namely 1a and 1b for fine-tuning of the network. The resultant CNN architecture is illustrated in Fig. 4. It incorporates three convolutional layers (conv1, conv2, and conv3), each followed by the ReLU activation function. All inputs are convolved with convolutional kernels with a stride of one to output feature maps. Two pooling layers (pool1 and pool2) used 2 × 2 kernels with a stride of one.

In our network for XMU dataset experiment, the conv1 layer has ten kernels of size $11 \times 11$; the conv2 layer has six kernels of size $5 \times 5$, and the conv3 layer has $23$ kernels of size $5 \times 5$. For XMUPlus dataset experiment, the CNN architecture is made up of 10 kernels of size $3 \times 3$; the conv2 layer has 11 kernels of size $7 \times 7$, and the conv3 layer has 22 kernels of size $3 \times 3$. In the FC layer, the network concatenated high-level features learned by the convolutional layers. Finally, a single neuron is computed as the final output of the network. In the last layer of CNN, each vehicle logo image is categorised with its feature vector by Softmax classifier. The specified hyper-parameters of selected CNN architecture for both XMU and XMUPlus experiments are tabulated in Table 3.

To further reflect the convergence of network, we have fine-tuned on both architectures 1a and 1b via increment of the number of CNN training epoch to become 50. We utilised the learning rate with values of 0.0027 and 0.0058 for experiments using XMU and XMUPlus datasets, respectively. As illustrated in Fig. 6, the objective or training loss converges at an earlier stage fifth epoch. The testing set is validated to obtain classification accuracy for every epoch. We have compared the classification accuracy of each epoch corresponds with its training time as demonstrated in Table 4. For the XMU dataset, the total training time of 50 epochs spends 21.8 min. In fact, the network begins to converge at the 10th epoch and achieve an accuracy of 99.5% in 4.35 min. This result has slightly outperformed the seminal work, which is utilising pretrained CNN approach [1] and XMU dataset, the reported accuracy is 99.07 with a training time of 15 min (without utilising GPU). On the other hand, the total training time of 50 epochs for XMUPlus dataset consumes 24.7 min. In addition, the accuracy of XMUPlus dataset experiment exceeds 99% after 30 epochs in about 15 min.

The results indicate that the proposed CNN framework with hyper-parameters optimisation by PSO is feasible to be implemented in ITS, more particularly, in the area of VLR. Table 5 shows a number of recent predominant approaches which are implemented in the area of VLR. The recognition accuracy of our proposed method outperforms than other handcrafted feature extraction methods. With the implementation of an efficient PSO approach, our proposed system is able to search and optimise the best combination of hyper-parameters for CNN architecture automatically.

In the aspect of computation time, the system we have presented should be conducive for a number of real-time

| Table 1 | PSO search space boundaries of hyper-parameter |
| Search boundaries | $\alpha$ | Conv1 | Conv2 | Conv3 |
| lower, Lower Boundary | $1 \times 10^{-5}$ | 1 | 3 | 3 | 1 |
| upper, Upper Boundary | $1 \times 10^{-2}$ | 10 | 11 | 9 | 20 | 9 | 30 |

| Table 2 | Selection of CNN architecture through multiple PSO experiments |
| Iteration number | Population size | XMU dataset | Maximum accuracy, % | PSO training time, h | XMUPlus dataset | Maximum accuracy, % |
| 10 | 10 | 4.14 | 1a | 99.53 | 4.87 | 1b | 98.10 |
| 10 | 15 | 6.15 | 2a | 99.73 | 7.35 | 2b | 98.51 |
| 10 | 20 | 8.20 | 3a | 99.80 | 9.75 | 3b | 98.82 |
| 15 | 10 | 5.90 | 4a | 99.80 | 6.97 | 4b | 98.15 |
| 15 | 15 | 5.58 | 5a | 99.60 | 10.36 | 5b | 98.46 |
| 15 | 20 | 12.11 | 6a | 99.73 | 14.38 | 6b | 99.18 |
| 20 | 10 | 7.73 | 7a | 99.80 | 9.50 | 7b | 99.23 |
| 20 | 15 | 11.74 | 8a | 99.80 | 14.84 | 8b | 99.18 |
| 20 | 20 | 15.73 | 9a | 99.80 | 18.68 | 9b | 99.54 |

The CNN architectures of 1a and 1b are in bold values to indicate that both CNN architectures are selected in classifying XMU and XMUPlus datasets.
The system is able to perform a hyper-parameters optimisation and CNN training process for each vehicle logo image in <5 h, without requiring parameter tuning and ad hoc tricks. This speed is adequate for applications such as traffic access and monitoring control when focusing on more than ten classes of vehicle manufacturer. As we compare our method with the PCA pretrained CNN [1] in terms of the training time, our proposed system needs only 4 h 8 min for hyper-parameters optimisation.

### Table 3
Result of the optimised structure of the proposed CNN for XMU and XMUPlus datasets experiments

<table>
<thead>
<tr>
<th>Layer</th>
<th>Parameter</th>
<th>Value (XMU)</th>
<th>Value (XMUPlus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>kernel size</td>
<td>11 × 11</td>
<td>3 × 3</td>
</tr>
<tr>
<td></td>
<td>#Kernels</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>stride</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>input size</td>
<td>70 × 70</td>
<td>70 × 70</td>
</tr>
<tr>
<td></td>
<td>output size</td>
<td>60 × 60 × 10</td>
<td>68 × 68 × 10</td>
</tr>
<tr>
<td>Pool1</td>
<td>kernel size</td>
<td>2 × 2</td>
<td>2 × 2</td>
</tr>
<tr>
<td></td>
<td>stride</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>output size</td>
<td>30 × 30 × 10</td>
<td>34 × 34 × 10</td>
</tr>
<tr>
<td>Conv2</td>
<td>kernel size</td>
<td>5 × 5</td>
<td>7 × 7</td>
</tr>
<tr>
<td></td>
<td>#Kernels</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>stride</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>output size</td>
<td>26 × 26 × 6</td>
<td>28 × 28 × 11</td>
</tr>
<tr>
<td>Pool2</td>
<td>kernel size</td>
<td>2 × 2</td>
<td>2 × 2</td>
</tr>
<tr>
<td></td>
<td>stride</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>output size</td>
<td>13 × 13 × 6</td>
<td>14 × 14 × 11</td>
</tr>
<tr>
<td>Conv3</td>
<td>kernel size</td>
<td>5 × 5</td>
<td>3 × 3</td>
</tr>
<tr>
<td></td>
<td>#Kernels</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>stride</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>output size</td>
<td>9 × 9 × 23</td>
<td>12 × 12 × 22</td>
</tr>
<tr>
<td>FC</td>
<td>output size</td>
<td>1863</td>
<td>3168</td>
</tr>
<tr>
<td>output</td>
<td>output size</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 4
Training process of CNN architecture using XMU and XMUPlus datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>XMU</th>
<th>XMUPlus</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) XMU dataset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>epoch</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>CNN training time, min</td>
<td>4.4</td>
<td>8.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>XMUPlus</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b) XMUPlus dataset</td>
<td></td>
</tr>
<tr>
<td>epoch</td>
<td>10</td>
</tr>
<tr>
<td>CNN training time, min</td>
<td>4.9</td>
</tr>
<tr>
<td>accuracy, %</td>
<td>98.10</td>
</tr>
</tbody>
</table>

Fig. 6 Training curve of using XMU dataset and XMUPlus dataset. The red solid line indicates the training loss and the blue dashed line indicates the validation loss. After five epochs, the blue dashed line lies above the red solid line which shows our approach is effective according to machine learning theory. The green dash-dotted line represents the accuracy.

Table 4: Training process of CNN architecture using XMU and XMUPlus datasets.
process; meanwhile, <5 min (10 epochs) for training of CNN architecture. Hence, it is expected that our proposed end-to-end CNN network system could be pretrained to significantly further improve both training procedure and recognition accuracy.

4 Conclusion
In this paper, we present a complete and efficient CNN regime with hyper-parameters optimisation by PSO for VLR. The trainable and stackable structure of CNN optimised by PSO exhibits the ability of extracting discriminative features from low-level to high-level features from the vehicle logo images. The experimental results have demonstrated that the proposed CNN system is able to achieve satisfactory results because of its robustness against various shifting, noise contaminations, and illumination conditions. The proposed approach has shown promising results toward bridging the gap between CNN hyper-parameters tuning and computational efficiency without sacrificing classification performance. The optimised and fine-tuned CNN model derives a more efficient image classification scheme than the manually tuned CNN. In future work, we will focus on more challenging training sets with complicated imaging conditions. Apart from that, more SI or hybrid methods can be considered as surrogate to further optimise the network, thus resulting in an even more optimised real-time vehicle recognition system possible.

5 Acknowledgments
The authors thank PLUS Malaysia, the builder of North–South Expressway (NSE), for providing real-time traffic data in this work. This work was financially supported by the Fundamental Research Grant Scheme (FRGS) from the Ministry of Higher Education of Malaysia with the grant no. FRGS/1/2017/TK04/UM/02/4.

6 References