Deep Learning-based Vehicle Recognition Schemes for Intelligent Transportation Systems

by

Xiren Ma

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Science degree

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the increasing highlighted security concerns in Intelligent Transportation System (ITS), Vision-based Automated Vehicle Recognition (VAVR) has attracted considerable attention recently. A comprehensive VAVR system contains three components: Vehicle Detection (VD), Vehicle Make and Model Recognition (VMMR), and Vehicle Re-identification (VReID). These components perform coarse-to-fine recognition tasks in three steps. The VAVR system can be widely used in suspicious vehicle recognition, urban traffic monitoring, and automated driving system. Vehicle recognition is complicated due to the subtle visual differences between different vehicle models. Therefore, how to build a VAVR system that can fast and accurately recognize vehicle information has gained tremendous attention.

In this work, by taking advantage of the emerging deep learning methods, which have powerful feature extraction and pattern learning abilities, we propose several models used for vehicle recognition. First, we propose a novel Recurrent Attention Unit (RAU) to expand the standard Convolutional Neural Network (CNN) architecture for VMMR. RAU learns to recognize the discriminative part of a vehicle on multiple scales and builds up a connection with the prominent information in a recurrent way. The proposed ResNet101-RAU achieves excellent recognition accuracy of 93.81% on the Stanford Cars dataset and 97.84% on the CompCars dataset. Second, to construct efficient vehicle recognition models, we simplify the structure of RAU and propose a Lightweight Recurrent Attention Unit (LRAU). The proposed LRAU extracts the discriminative part features by generating attention masks to locate the keypoints of a vehicle (e.g., logo, headlight). The attention mask is generated based on the feature maps received by the LRAU and the preceding attention state generated by the preceding LRAU. Then, by adding LRAUs to the standard CNN architectures, we construct three efficient VMMR models. Our models achieve the state-of-the-art results with 93.94% accuracy on the Stanford Cars dataset, 98.31% accuracy on the CompCars dataset, and 99.41% on the NTOU-MMR dataset. In addition, we construct a one-stage Vehicle Detection and Fine-grained Recognition (VDFG) model by combining our LRAU with the general object detection model. Results show the proposed VDFG model can achieve excellent performance with real-time processing speed. Third, to address the VReID task, we design the Compact Attention Unit (CAU). CAU has a compact structure, and it relies on a single attention map to extract the discriminative local features of a vehicle. We add two CAUs to the truncated ResNet to construct a small but efficient VReID model, ResNetT-CAU. Compared with the original ResNet, the model size of ResNetT-CAU is reduced by 60%. Extensive experiments on the VeRi and VehicleID dataset indicate the proposed ResNetT-CAU achieve the best re-identification results on both datasets. In summary, the experimental results on the challenging benchmark VMMR and VReID datasets indicate our models achieve the best VMMR and VReID performance, and our models have a small model size and fast image processing speed.

Acknowledgements

First of all, I would like to give my sincere appreciation to my supervisor Professor Azzedine Boukerche. He offered me the opportunity to pursue my graduate study at the University of Ottawa and provided me the financial support for the research work. His valuable and constructive suggestions for my research work made this possible.

Second, I would like to give my special thanks to Dr. Peng Sun, who provided me valuable suggestions on my research work and thesis writing. His patient guidance has helped me overcome many challenges.

I also want to extend my thanks to all the members of the PARADISE Laboratory. Thanks to Abdul Jabbar Siddiqui and Yiheng Zhao, whose valuable suggestions helped me start my research work. I also feel incredibly grateful to my friends Peili Xu, Yufan Zhou, Jiahao Wang, Mingzhi Sha, Qiyue Wu, Zhiqing Tu, etc., for their friendship and encouragement.

Finally, I would like to thank my parents for encouraging and supporting me over these years. Their love and patience helped me finish my graduate study.

Publication

- **Xiren Ma** and Azzedine Boukerche, "An AI-based Visual Attention Model for Vehicle Make and Model Recognition", in *IEEE Symposium on Computers and Communications (ISCC), pp. 1-6, doi: 10.1109/ISCC50000.2020.9219660, 2020*
- Azzedine Boukerche and **Xiren Ma**, "Vision-based Autonomous Vehicle Recognition: A New Challenge for Deep Learning-based Systems", *ACM Computing Surveys 54, 4, Article 84 (April 2021), 37 pages. DOI:https://doi.org/10.1145/3447866*
- Azzedine Boukerche and **Xiren Ma**, "A Novel Smart Lightweight Visual Attention Model for Fine-Grained Vehicle Recognition", submitted to *IEEE Transactions on Intelligent Transportation Systems (T-ITS), 2020*
- **Xiren Ma** and Azzedine Boukerche, "An Efficient Real-Time Vehicle Re-Identification Scheme Using Urban Surveillance Videos", in *IEEE International Conference on Communications (ICC), 2021*

Table of Contents

List of Tables

List of Figures

Nomenclature

BN Batch Normalization

CNN Convolutional Neural Network

GRU Gate Recurrent Unit

ITS Intelligent Transportation System

LPR License Plate Recognition

LSTM Long-Short-Term Memory

MLP Multilayer Perceptron

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

UAV Unmanned Aerial Vehicle

VAVR Vision-based Automated Vehicle Recognition

VD Vehicle Detection

VDFR Vehicle Detection and Fine-grained Recognition

VMMR Vehicle Make and Model Recognition

VReID Vehicle Re-identificaiton

Chapter 1

Introduction

1.1 Background and Motivation

The development of computer vision technology, the widely used surveillance cameras, and onboard cameras have provided significant advantages for implementing vision-based Intelligent Transportation System (ITS) and Smart City applications such as the autonomous driving system and the traffic management system [4, 266, 189, 194, 86, 289, 73, 267, 74, 208, 205, 268, 130, 269, 242, 16, 10, 23, 19, 39, 21, 22]. Since vehicles are the main participants in the transportation system, capturing the vehicle-related information is essential to traffic management (e.g., calculating the travel time of the vehicles is useful for traffic control) and public security (e.g., fast locating the suspicious vehicle).

Vehicle recognition has attracted widespread attention, and many researchers, companies, and governments have invested money, time, and energy to develop Vision-based Automated Vehicle Recognition (VAVR) systems [260, 229, 180]. A complete VAVR system contains three components: Vehicle Detection (VD), Vehicle Make and Model Recognition (VMMR), and Vehicle re-identification (VReID). These components perform coarse-to-fine recognition tasks in three steps. A coarse-to-fine procedure of recognizing vehicle information of VAVR is shown in Figure 1.1. First, the VD module receives an image captured by the camera (onboard or surveillance camera) as input. The module will detect the vehicles in the image and identify the vehicle type such as sedan, hatchback, SUV, van, etc. Second, the detected vehicles are cropped from the original image, and then forwarded to the VMMR module to distinguish the detailed information, such as vehicle make, model, and even the year of manufacture (e.g., Audi Q8 2018, Ford Mondeo 2017). If we are searching for a vehicle in the recorded video, which contains a massive amount of vehicles, the VD and the VMMR modules will filter out a considerable number of vehicles to narrow the search space, and the remaining vehicle images are the candidate vehicles. Then, the VReID module is adopted to identify and verify the target vehicle from the candidate vehicles. Compared with VMMR, VReID goes one step further. It requires subtle differences among vehicles of similar appearance to be distinguished.

An efficient VAVR system is very important for ITS and Smart City. For example, when a vehicle is stolen or grabbed by a criminal, the police usually need to set up inspection

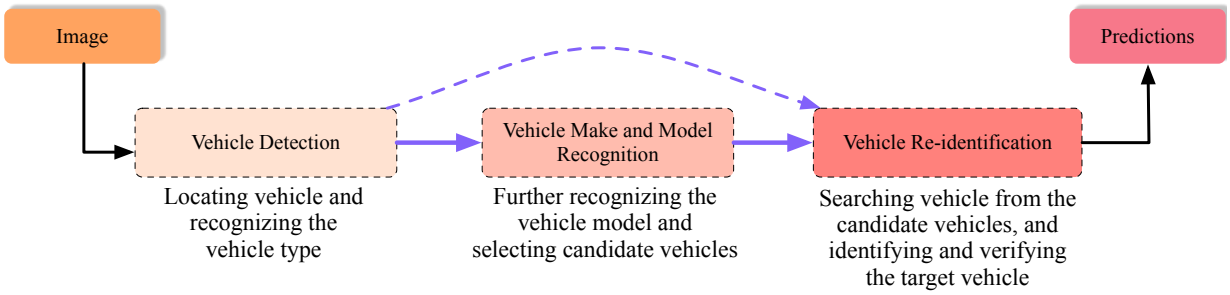


Figure 1.1: Overview of the coarse-to-fine vehicle recognition procedure. The dashed line represents the VReID module can directly find the target vehicle from the vehicle candidates extracted by the VD module.

posts to intercept passing vehicles or manually search for a vehicle with a specific type, make and model through the surveillance camera pre-deployed around the city, which takes much time and is inaccurate in the end. With the help of the VAVR system the vehicle that is stolen or grabbed by the criminal can be fast located and tracked. By deploying a small VAVR system on the vehicles such as Unmanned Aerial Vehicle (UAV) and car. The vehicle can directly process the images captured by the onboard camera. Then, relying on the vehicular network and the vehicle-to-everything techniques [225, 144, 164, 200, 262, 253, 36, 34, 66, 38, 33, 41, 97, 35, 217, 65, 106, 6, 45, 69, 70, 44, 81, 98, 57, 64, 219, 42, 46, 67, 54, 83, 59, 51, 116, 76, 52, 58, 226, 100, 47, 218, 53, 203, 5, 37, 95, 103, 196, 29, 78, 207, 114, 75, 68, 283, 105, 30, 79, 84, 61, 101, 63, 56, 48, 77, 115, 40, 50, 72, 55, 99, 26, 113, 204, 85, 231, 220, 87, 62, 71, 195, 49, 254, 123, 90, 80, 43, 198, 7, 13, 239, 12, 8, 241, 20, 240, 60, 9, 15, 14, 18, 11, 24, 17, 25, 109, 274], the acquired vehicle information can be fast transmitted to the target agent (e.g., traffic control center, police station), which will significantly help make a fast decision and improve the regional security.

Traditional vehicle recognition mainly relies on License Plate Recognition (LPR) [110]. The vehicle information can be checked through the license plate. Since the license plate is the unique ID of a vehicle, the registration center records the detailed vehicle information related to the license plate. However, LPR has several limitations. First, in some security-related situations, the license plate is often damaged, obscured, removed, or even faked. Second, recognizing license plate information requires a high-resolution image. In addition, changes in lighting and various camera angles may reduce the accuracy of LPR. The VAVR system can be used as a complementary solution to the LPR system. When license plate information is not available, the VAVR system is able to directly recognize the vehicle information according to the appearance feature. We can also cross-check the recognition results from the VAVR system and the LPR system to improve recognition accuracy, and prevent license plate fraud [82]. Overall, the VAVR system could save a lot of resources, time, and workforce.

Vision-based vehicle recognition can be considered a challenging multi-class image classification problem. The challenges and issues are summarized as follows:

1. Inter-class Similarity and Intra-class Variance: Different classes of vehicles could have very similar appearances, while vehicles of the same class may differ significantly.

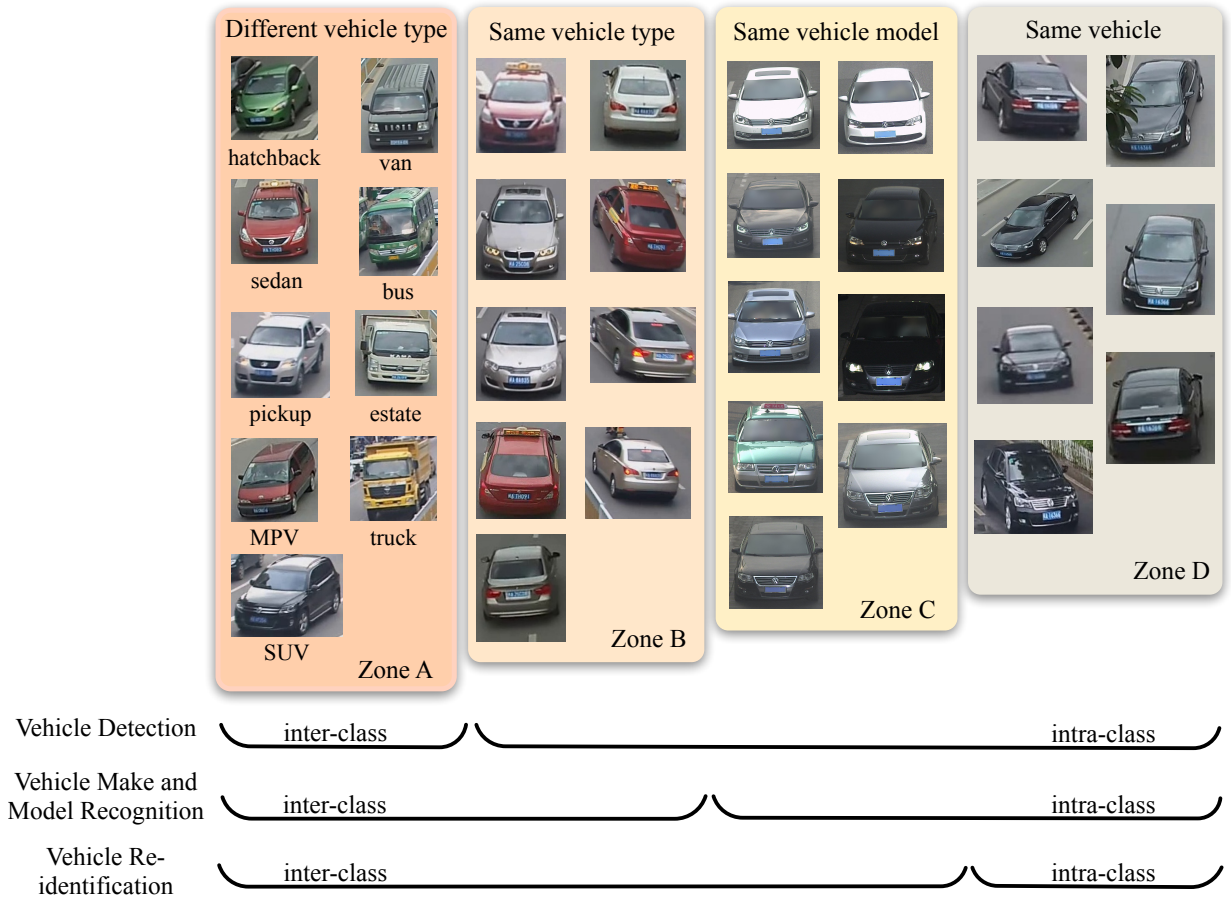


Figure 1.2: Illustration of coarse-to-fine inter-class similarity and intra-class variance of vehicle recognition. Zone A shows different vehicle types. Different vehicles can have a similar appearance, for example, truck and estate. Zone B shows vehicles of the same type (sedan). The appearance and shape of these vehicles are quite different. Zone C shows vehicles of the same model. The appearance of these vehicles varies widely depending on color and illumination. Zone D shows the same vehicle from different viewpoints. Humans cannot guarantee that these images belong to the same vehicle.

2. Environmental Conditions: Different environmental conditions would significantly affect the model performance, such as weather conditions (e.g., snowy, foggy, sunny, rainy), road conditions (e.g., highway, urban road), illuminations, and occlusions.

3. Modified Appearance: The original vehicle appearance can be modified, for example, adding bumpers, adding spoilers, drawing other patterns on the hood or trunk, placing decorations in front of the window.

4. Diversity: There are a massive number of vehicle types, makes and models. Every vehicle has diversified colors or decorations.

5. Camera-related Issue: Low-resolution camera, camera refresh rate, different camera angles, and image distortion caused by the camera.

Table 1.1: The Range of Inter-class and Intra-class of VD, VMMR, and VRe-ID

Task	Inter-class	Intra-class
VD	Different vehicle types (different objects)	Different make and models within the same vehicle type
VMMR	Different vehicle makes and models	Different vehicles (varying in decoration and color) within the same vehicle make and model
VReID	Different vehicles	Different viewpoints of the same vehicle

6. Dataset: Deep learning is a data-driven approach, and the deep learning-based model requires a suitable (large-scale, coverage of different situations) dataset to train a model for each task.

7. Combining Additional Information: How to combine different information (e.g., license plate information, lidar information) with the appearance feature is a challenge.

Among these common challenges and issues, the inter-class similarity and intra-class variance are the major problems that exist in VD, VMMR, and VReID. The range of the inter-class and intra-class for each task is illustrated in Table 1.1. An illustration of inter-class similarity and intra-class variance of vehicle recognition is shown in Figure 1.2. For example, in terms of VD, we can observe the inter-class similarity from zone A, and the intra-class variance from Zone B to D. To solve the inter-class similarity and intra-class variance, one solution is to pay more attention to the discriminative part of the vehicle (e.g., headlight, logo). Relying on the discriminative local features, the vehicle can be distinguished from other vehicles.

The rapid development of the Convolutional Neural Network (CNN) has helped many models achieve great performance on VMMR. Recent studies [31, 184, 278] indicate that learning the features of the discriminative part of an object (e.g., the tail of birds, the head of dogs, the headlight of vehicles) can improve the fine-grained recognition performance. Some studies [278, 277] address fine-grained recognition problem by training the object detector (e.g., R-CNN [124] and Fast-RCNN [125]) as the part detector to locate the part regions. Then, these semantic part regions are cropped out and sent to the feature extraction network to extract discriminative features. However, there are several drawbacks of these methods: (1) These part detector-based methods often require part bounding box annotations to train the part detectors. However, the part bounding box annotation requires expert knowledge and labeling the part is time-consuming. Only a few datasets provide such detailed annotations, which may limit the applications of these methods. (2) The human-defined part labels are not often well-defined, which may introduce noise and affect the performance of the model, resulting in unable to achieve the optimal result. (3) The part detector and the feature extraction network usually need to be trained separately, so these models can not be trained end-to-end.

Humans recognize the object by glimpsing different parts of the object [108]. Inspired by the human attention mechanism, the feed-forward attention methods [141, 222] were proposed to implicitly find the part regions (without cropping) to improve the model performance. These models adopt different strategies to generate the attention mask,

which highlights the prominent regions of the image. By multiplying these attention masks with the feature maps of the object, the part features are implicitly extracted. However, these methods usually ignore the coarse-to-fine relationship of the part features. They either only extract one-scale discriminative features, or extract multi-scale features but do not combine them. For instance, the Spatial Weighted Pooling (SWP) [141] method only extracts discriminative features from the feature maps generated by the last convolutional layer.

Modeling the coarse-to-fine relationship of the part features is helpful for recognizing an object, especially the fine-grained information of the object. For example, if we stand very close to an object, we can only observe a small part of the object, such as the foot of a tiger (fine-grained feature). If we stand far away from an object, we cannot determine whether it is a tiger or another object only based on its shape (coarse feature). Only when we combine and model the relationship of the coarse feature and the fine-grained feature (e.g., comparing the foot and the shape of the animal) can we determine the category of the object (especially the fine-grained category).

Based on the observations, we can build a unit to focus on extracting the discriminative local features and model the coarse-to-fine relationship of local features. Then, by combining the unit with a global feature extraction network, a powerful model can be constructed.

1.2 Problem Statement

Generally, when an image contains multiple vehicles, we usually need to use a VD model to detect these vehicles and crop them out according to each vehicle's predicted bounding box. Then, the cropped vehicle images are sent to the VMMR or VReID model. For the VMMR task, our work mainly focuses on recognizing fine-grained information about a vehicle. For the VReID task, our work mainly focuses on comparing the similarities of different vehicles and find the target ones. For the VDFR task, our model can locate and recognize the fine-grained information of the vehicle and the major goal is still recognizing the fine-grained information. In addition, the VDFR model can easily detect multiple vehicles occur in the image by training it with a suitable dataset. Therefore, we assume that the image received by the VMMR, VDFR, and VReID model contains only one vehicle. Besides, our VMMR and VReID models have the ability to batch process images. Therefore, they can simultaneously identify the information of multiple vehicles. For a complete VAVR system, any robust real-time vehicle detection model can combine with our models.

1.3 Methodology

Based on the recently proposed Prototype Recurrent Unit (PRU) in [181], we build the Recurrent Attention Unit (RAU). PRU was proposed as a prototypical example for future study of LSTM-like recurrent networks [181]. PRU has a compact structure and captures

the key component of LSTM and GRU [93]. PRU can only process the one-dimensional sequence vector. As an improvement, our RAU can directly process the feature maps extracted by the convolutional layers. By adding RAUs to ResNet, we proposed two models for VMMR.

To reduce the model size of RAU, we simplify its structure and propose the Lightweight Recurrent Attention Unit (LRAU). Then, we add LRAU to ResNet, VGG16, InceptionV3, obtaining several models for VMMR. In addition, the VMMR method usually receives an image that contains one vehicle and only outputs the fine-grained vehicle information. However, the location and size information of the vehicle is often ignored. To detect vehicles and recognize their fine-grained information simultaneously, we incorporate LRAU with an object detection method called Single Shot Detector (SSD) [173], and then obtained a one-stage Vehicle Detection and Fine-grained Recognition (VDFR) model. The proposed SSD-LRAU can generate the bounding box and fine-grained vehicle information simultaneously.

For the VReID task, we want to construct a model with excellent local feature extraction ability while maintaining a small model size. Therefore, we further simplify the structure of the attention unit and proposed the Compact Attention Unit (CAU). Then, we add it to a truncated ResNet, obtaining ResNetT-CAU.

1.4 Contributions

The major contribution of our work can be concluded as follows:

1. We propose the Recurrent Attention Unit (RAU), which is a modular attention unit that can be applied on different layers of standard CNN architectures to extract prominent information from different scales. Each RAU takes the feature maps generated by the convolutional layer and the attention state generated by the previous RAU as inputs. Afterward, the unit produces a new state for the next RAU. The mechanism is not only extracting discriminative information from different layers but also combining them. This process allows the model to recognize an object by evaluating the discriminative features of different resolutions. In addition, RAU can locate the local regions without using any part annotations. To verify the effectiveness of RAU, we add the proposed RAUs to the standard CNN architectures to solve the VMMR task. The proposed ResNet101-RAU achieves an excellent recognition accuracy of 93.81% on the Stanford Cars dataset, 97.84% on the CompCars dataset, and 98.90% on the CompCars Surveillance dataset.

2. To further improve the local feature extraction of RAU and build an efficient VMMR model, we optimize the structure of RAU and propose a Lightweight RAU (LRAU). Then, LRAU is added to ResNet, VGG and InceptionV3 to build three lightweight VMMR models. Our models achieve the state-of-the-art results with 93.94% accuracy on the Stanford Cars dataset, 98.31% accuracy on the CompCars dataset, and 99.41% on the NTOU-MMR dataset. In addition, the VMMR method usually receives an image that contains one vehicle and only outputs the fine-grained

vehicle information. However, the location and size information of the vehicle is often ignored. To detect vehicles and recognize their fine-grained information simultaneously, we incorporate LRAU with an object detection method called SSD, and then obtained a one-stage Vehicle Detection and Fine-grained Recognition (VDFR) model. The proposed SSD-LRAU can generate the bounding box and fine-grained vehicle information simultaneously. Comparing with SSD, the number of parameters of SSD-LRAU is reduce by 82%, while achieving a better result.

3. Furthermore, to solve the VReID task, we proposed a Compact Attention Unit (CAU), which extracts the discriminative local features without using any part annotation. The CAU has a simple structure and is easy to be added to different standard CNN architectures. The CAU uses an attention map generated by the convolutional layer to select the discriminative regions from the feature maps generated by CNN. The proposed attention model can effectively locate the discriminative regions without using any part annotation. By adding multiple CAUs to a truncated ResNet, we construct an effective VReID model. The model extracts both the global feature and discriminative local feature and uses them to identify the target vehicle. More importantly, the model has a small model size and fast image processing speed (5.1 ms per batch), making the model more suitable for smaller devices and the real-time environment.

4. To evaluate the proposed models, we conduct extensive experiments on several benchmark datasets to explore the optimal parameter settings for our proposed models and compare the results with many state-of-the-art methods. We also evaluate the model performance under different environmental conditions to verify the robustness of our proposed models.

1.5 Thesis Outline

- Chapter 2 starts with introducing the fundamental knowledge of CNN. Then, the commonly adopted deep learning backbones are introduced accordingly. Then, a comprehensive literature review is provided to introduce the classification and development of the deep learning-based methods for VD, VMMR, and VReID. Finally, the commonly adopted benchmark datasets and open challenges are discussed in detail.

- Chapter 3 introduces the proposed RAU. The model structure and the combination with ResNet are explained in detail. Then, we conduct the ablation study and comparison experiments on the benchmark VMMR datasets.

- Chapter 4 presents the LRAU. Then, we introduce the proposed VMMR model and the VDFR model. We systematically evaluate the models by conducting a series of comparison experiments.

- Chapter 5 proposes the CAU. We combine CAU with a truncated ResNet, constructing an efficient VReID model. The model performance is evaluated by conducting experiments on two benchmark VReID datasets.

- Chapter 6 summarizes the work we have done in this thesis and presents future research directions for vehicle recognition study.

Chapter 2

Related Work

In this chapter, we first introduce the fundamental knowledge of CNN. Then, we review the commonly adopted CNN architectures. Then, we systematically review the classification and development of the deep learning-based methods for VD, VMMR, and VReID. We also present a detailed introduction to different vehicle recognition datasets used for a comprehensive evaluation of the proposed models. Last but not least, we critically discuss the major challenges and future research trends involved in each task.

2.1 Neural Network

The Neural Network (NN) is modeled as the acyclic computation graph representing the composition of a set of functions. The basic unit in a NN is the neuron, often called a node or unit. It receives input from some other nodes and computes an output. The computation of a typical neuron is the weighted sum of the inputs and is followed by a nonlinear activation. For NNs, the most common layer type is the fully-connected (FC) layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections.

A NN can be constructed by simply stacking multiple FC layers. For example, a Multilayer Perceptron (MLP) is a deep NN that contains at least three layers of nodes (two FC layers): an input layer, a hidden layer, and an output layer. A 2-layer MLP can be seen in Figure 2.1. The computation at the lth hidden layer is:

$$H^l = g(W^l H^{l-1}), \tag{2.1}$$

where W^l is a weight matrix, g is the activation function (will be introduced in section 2.1.2). Suppose we have a grey image $x \in \mathbb{R}^{1\times h\times w}$, which contains an animal, where h and w is the height and width of the image. A 2-layer MLP can be used to classify whether the animal is the cat or the dog. First, we reshape the input image to a column vector $x \in \mathbb{R}^{hw\times 1}$, then we set $W^{(1)} \in \mathbb{R}^{m\times hw}$ and $W^{(2)} \in \mathbb{R}^{2\times m}$, where m represents the number of hidden units. Finally, a 2-layer MLP can be expressed as follows:

$$f(x) = g(W^{(2)} \cdot g(W^{(1)}x)), \tag{2.2}$$

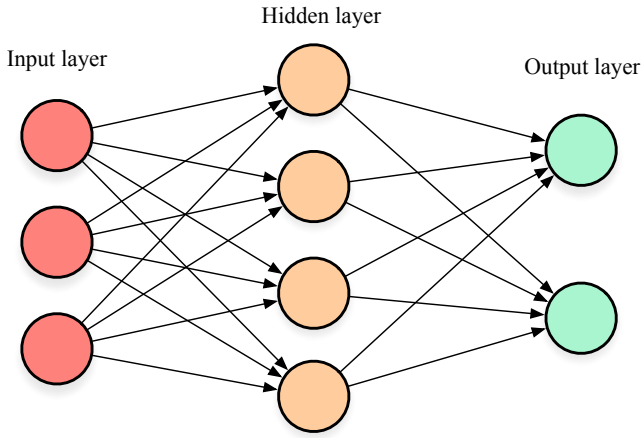


Figure 2.1: A 2-layer MLP (one hidden layer of 4 neurons(units) and one output layer with 2 neurons.

where $f(x) \in \mathbb{R}^{2\times 1}$ is the column vector that shows the scores of the image belonging to each category. $f(x)$ is often normalized by the softmax function or sigmoid function to generate the classification probabilities. In a CNN, the FC layer is often used to construct the classification network.

2.1.1 Convolutional Neural Network

CNN is very similar to ordinary NN. CNN can directly process the image with size $d\times h\times w$, where d is the number of channels. A CNN usually consists of the feature extraction network and classification network. The feature extraction network consists of the convolutional layer, pooling layer, activation layer, and normalization layer.

The convolutional layer is made up of a set of filters that have learnable weights and biases. Each filter usually has a small kernel size, i.e., 3×3 or 5×5, and extends through the full depth of the input. During the forward pass, the filters slide across the height and width of the input and perform a dot product between the filter and the corresponding region of the input. This operation is similar to the convolutional operation in signal processing and produces a m-dimensional feature map, m is the number of filters. For example, let $W \in \mathbb{R}^{d\times f_h\times f_w}$ be the filter, and $x \in \mathbb{R}^{d\times h\times w}$ be the input, and we set the stride size to 1 and padding size to 0. Then the output feature map of the convolutional layer will be of size $(h - f_h + 1) \times (w - f_w + 1)$. The value y_{ij} in the feature map can be formulated as follows:

$$y_{ij} = \sum_{c=0}^{d-1}\sum_{a=0}^{f_h-1}\sum_{b=0}^{f_w-1} W_{cab} \cdot x_{c(i+a)(j+b)}. \tag{2.3}$$

An overview of the convolution procedure is shown in Figure 2.2.

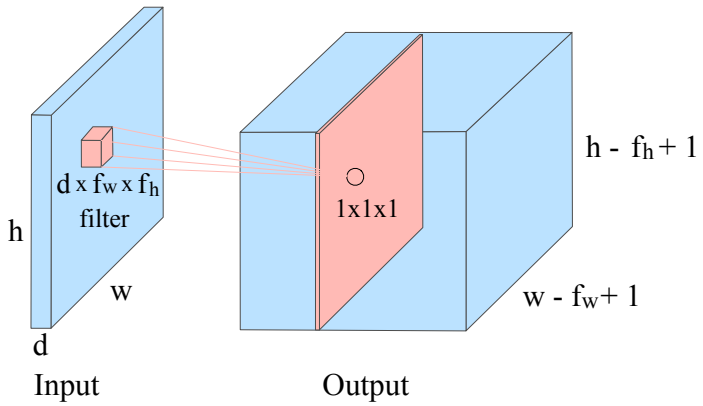


Figure 2.2: The convolution process.

2.1.2 Activation Layer

The activation layer takes the output of the convolutional layer as the input and performs some mathematical operations on the input. The activation function introduces non-linearity to the network. There are several commonly used activation functions:

1. Rectified Linear Unit (ReLU) [201] is often used in CNN. It computes the function $f(x) = max(0, x)$.
2. Leaky ReLU [187] attempts to fix the "dying ReLU" problem (gradient vanishing), and it computes $f(x) = \mathbb{1}(x < 0)(ax) + \mathbb{1}(x >= 0)(x)$, where α is a small constant.
3. $Tanh$ is another activation function and it is often used in Recurrent Neural Network (RNN). It computes $f(x) = tanh(x)$.
4. The sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ and softmax function $f(x)_c = \frac{e^{x_c}}{\sum_i^k e^{x_i}}$ are often used after the classification layer to normalize the output.

2.1.3 Pooling Layer

The pooling layer is often used in CNN to progressively reduce the spatial size of the feature map, hence, reduce the amount of the parameters and computation in the network. It also helps to control overfitting. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The output size is controlled by the filter size, stride size, and padding size. The most common approach used in pooling is the max pooling and average pooling. Figure 2.3 shows the result of max pooling using the filter of size 2 $\times$ 2, stride 2.

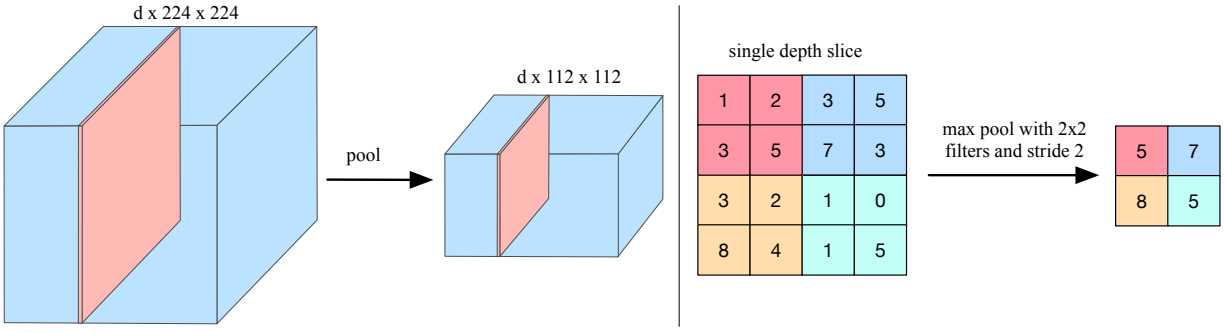


Figure 2.3: The process of downsampling the feature maps by the pooling layer. Left: The input feature maps ($d \times 224 \times 224$) are pooled with filter of size 2×2, stride 2. The size of the output is $d \times 112 \times 112$. Notice that the number of channels of the output remains the same as the input. Right: Max pooling operation on a depth slice by using a filter of size 2×2 with a stride of 2.

2.1.4 Batch Normalization

Batch Normalization (BN) is proposed by Loffe and Szegedy [145]. The method was initially proposed to solve the internal covariate shift. Generally, the output of an activation layer can be normalized by subtracting the batch mean and dividing by batch standard deviation. However, simply normalizing the input of each layer may change what the layer originally represents. To recover the presentation power of each layer and control the shift and scale level by the model itself. BN adds two trainable parameters to retain the training effect of each layer. In addition, this method allows us to train the model with a higher learning rate and reduce the dependence on weight initialization and image preprocessing, and it also provides some regularization. BN is formulated by the following functions:

$$\begin{aligned} \mu\mathbb{B} &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_{\mathbb{B}}^2 &\leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu\mathbb{B})^2 \\ \hat{x}_i &\leftarrow \frac{x_i - \mu\mathbb{B}}{\sqrt{\sigma_{\mathbb{B}}^2 + \epsilon}} \\ y_i &\leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \end{aligned} \tag{2.4}$$

where $x_i \in B = \{x_1, ..., x_m\}$ is the input in a mini-batch. γ and β are the learnable parameters.

2.1.5 Loss Function

To evaluate the model during the training process, we need a loss (cost) function. The loss function in a supervised learning problem measures the compatibility between a prediction

and the ground truth label. In a training batch, the loss is the average of the losses for the inputs. Let $s = f(x)$ be the score function of a NN, which takes the image x as input and outputs scores for each class. Then, the cross-entropy loss function is formulated as follows:

$$L = -\sum_{i=1}^{c} \hat{y}_i log(y_i), \tag{2.5}$$

$$y_i = \frac{e^{s_i}}{\sum_j e^{s_j}}, \tag{2.6}$$

where c is the number of classes, $\hat{y}$ is the ground truth vector for the input, and it is often one-hot encoded (1 for the ground truth category, 0 otherwise). s_i is the predicted score for class i, and it is often normalized by the softmax function to guarantee the score is between 0 and 1 and sum to 1. The loss function can be different when training a model for different tasks.

2.1.6 Optimization

The loss function is used to evaluate the performance of the model of a set of parameters θ. The goal of the optimization is to find θ that minimizes the loss. To optimize a function $s = f(x, \theta)$, the most original method is to randomly select several θ, and then select the one with the smallest loss value. Another method is to start with a random θ, and then randomly choose an update direction (perturbation) $\delta\theta$. When the selected perturbation minimizes the loss, we can update the parameter $\theta = \theta + \delta\theta$. However, this method is slow and computationally expensive, especially for a large model. An efficient way is to calculate the gradient of the loss function with respect to each parameter. Then, we can update the parameter according to the gradient.

When training the model, we repeatedly calculate the gradient of the loss function with respect to each parameter and update the parameter. This method is called gradient descent. Gradient descent is a general way to optimize a deep learning model. For a deep learning model, to make the model have a good generalization ability, we often use a large scale dataset (from thousands to millions of samples) to train the model. Hence, it is wasteful to compute the full loss function over the entire training set to perform only a single parameter update. It is also unrealistic due to the memory constraints of the computer. Therefore, a common approach is Stochastic Gradient Descent (SGD), which computes the gradient over randomly selected batches of the training data and updates the model's parameters by a small step:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta J(\theta_t) \tag{2.7}$$

where $J(\theta_t)$ is the objective function parameterized by a model's parameters θ. The learning rate α determines the size of the steps we take to reach a minimum.

There are several challenges that need to be addressed when optimizing a model:

- Choosing a proper learning rate can be difficult. A learning rate that is too small leads to slow convergence, while a large learning rate can hider convergence and cause the loss function to fluctuate around the minimum or even to diverge. Therefore, a practical way is to choose a large learning rate at the beginning and then reduce the learning rate at some point.

- Learning rate schedule can be hard. Learn rate schedules try to adjust the learning rate during training, i.e., reducing the learning rate according to a pre-defined schedule or when the change in objective between epochs falls below a threshold. However, these schedules and thresholds have to be defined in advance and can only be decided according to the training experience.

- Another key challenge of minimizing the loss function of the neural network is avoiding getting trapped in their numerous sub-optimal local minima. Dauphinet al. [104] argued that the difficulty arises not from local minima but saddle points, i.e., points where one dimension slopes up and other slopes down. These saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape, as the gradient is close to zero in all dimensions.

The descending direction of SGD oscillates a lot between each batch, so it usually takes a long time to find the minimum. Therefore, Momentum [211] is introduced to improve SGD:

$$\begin{aligned} v_{t+1} &= \rho v_t + \nabla_\theta J(\theta_t), \\ \theta_{t+1} &= \theta_t - \alpha v_{t+1}, \end{aligned} \tag{2.8}$$

where ρ is the coefficient of the momentum v. Instead of only using the gradient of the current step to guide the search, momentum accumulates the gradient of the past steps to determine the update direction. There are several variations of gradient descent algorithm such as Adagrad [112] and Adam [154]. A detailed comparison of these optimization algorithms can be found in Reference [223].

2.1.7 Backpropagation

The backpropagation algorithm works by calculating the gradient of the loss function with respect to each parameter by the chain rule. During backpropagation, the gradient of each layer (local gradient) will also be calculated, which avoids the redundant calculation of the middle terms in the chain rule.

Suppose we have a one-layer neural network, $f = x \cdot W$ is the FC layer, $y = \sigma(f)$ is the sigmoid activation function, $L = \frac{1}{2}(y - \hat{y})^2$, is the suqare error loss function. Both the x and W are vectors. Then the gradient of the weight W can be represeted through chain rule:

$$\frac{dL}{dW} = \frac{dL}{dy} * \frac{dy}{df} * \frac{df}{dW} = x^T * f\sigma(1 - f) * (y - \hat{y}), \tag{2.9}$$

with the same idea, the gradient of x can be computed as:

$$\frac{dL}{dx} = \frac{dL}{dy} * \frac{dy}{df} * \frac{df}{dx} = (y - \hat{y}) * f\sigma(1 - f) * W^T. \tag{2.10}$$

For a multi-layer neural network, it is hard to directly compute the gradient of each weight. With the chain rule, we only need to calculate the local gradient of each layer and multiply it with the gradient with regard to its output. This is very helpful, especially when the neural network is very large.

2.2 Commonly Adopted Backbone Networks

In this article, we mainly introduce the deep learning-based vehicle recognition method, which primarily relies on CNN. CNN has a great ability to understand image content and has proven its superiority in many research areas such as classification, segmentation, detection, and re-identification [174, 238, 152]. The general CNN architecture consists of the convolutional layer, activation layer, pooling layer, and FC layer. A feature extraction network is constructed by stacking multiple convolutional layers, activation layers, and pooling layers. The feature extraction network extracts the features of the input image. Then, the features are sent to the classification network to generate the classification probabilities for each category. Generally, the classification network is the FC layer, usually followed by a softmax function or a sigmoid function to normalize the generated scores.

Recently, many works have been proposed to improve CNN. The improvement of CNN includes: choosing different filter sizes for the convolutional layer, increasing the depth and width of CNN, and adopting different strategies to stack the convolutional layer. There are many famous CNN architectures such as LeNet5 [161], AlexNet [158], VGG [233], GoogLeNet [244], ResNet [134] and DenseNet [142]. The characteristics of these CNN architectures are introduced below:

- LeNet5 [161] uses five convolutional layers to progressively aggregate simple features into complicated features. This network is the starting point for many CNN architectures.
- AlexNet [158] has five convolutional layers and three FC layers. In the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 [224] competition, AlexNet achieved a classification test error rate of 15.3%, which is 10.9% lower than the second-best method. CaffeNet[149] has the same structure as AlexNet, only the position of the pooling layer and normalization layer is different.
- VGG [233] is a popular backbone network. The authors increased the depth of the network to enhance the feature extraction ability of the network. Unlike LeNet and AlexNet, which stack filters of different kernel sizes, VGG mainly uses filters with a kernel size of 3×3.
- GoogLeNet [244] is another popular backbone network. It uses a special structure called Inception. In the Inception module, features are processed simultaneously by filters of size 1×1, 3×3, and 5×5. The outputs of filters have the same spatial size and are merged by concatenating along the depth dimension. The 1×1 filter is mainly used to reduce the feature dimension to control the computational complexity.

Table 2.1: Summary of Commonly Adopted CNN Architectures

Model	Characteristics
LeNet5 (1998) [161]	Five convolutional layers, one of the earliest CNNs
AlexNet (2012) [158]	Five convolutional layers and three FC layers
VGG (2014) [233]	Filters with a kernel size of 3×3 were mainly used
GoogLeNet (2014) [244]	Inception module (stacking filters of size 1×1, 3×3, and 5×5)
ResNet (2015) [134]	Residual module (shortcut connection between convolutional layers)
DenseNet (2016) [142]	Dense connection among layers

By stacking the Inception module, more convolutional layers can be used to generate high-dimensional semantic features.

GoogLeNet and the VGG net were the first and second winners of the ILSVRC2014 competition. Their authors both adopted the idea of carefully increasing the depth and the width of the network to improve the model recognition accuracy. They revealed the network depth is important for CNN. However, by simply stacking more layers into the network, a notorious problem of gradient vanishing or exploding occurs. Also, the accuracy could saturate and then degrade rapidly, even if the very deep CNN starts converging (degradation problem). ResNet [134] and DenseNet [142] successfully solved the problem mentioned above:

- ResNet was proposed in Reference [134]. The core component of ResNet is the Residual module. The authors built a shortcut among the convolutional layers. Therefore, the gradient can be better backpropagated to the shallow layers. Consequently, with the help of the Residual module, ResNet152 stacks 152 convolutional layers, and its model size is still smaller than VGG [134].
- DenseNet [142] has a dense connectivity pattern. In the lth layer, it has l inputs. The inputs are from all the preceding convolutional blocks and are combined by concatenating. The generated feature maps are passed on to all subsequent $L - l$ layers. There are $\frac{L(L+1)}{2}$ connections in a L-layer network. These dense connections improve the gradient backpropagation and allow the reuse of features from different layers, making the network easier to train. The model size is also reduced, and the overfitting is effectively suppressed.

The CNN architectures mentioned above were originally proposed for image classification. Researchers found that these models have exceptional abilities in feature extraction. Therefore, these models are often used as the backbone network for different tasks. We summarize the characteristics of these commonly adopted backbones in Table 2.1.

In the following sections, we provide a systematic review of the works that addressing different vehicle recognition tasks, i.e., VD, VMMR, VReID. An overview of the classification of deep learning-based vehicle recognition models is shown in Figure 2.4.

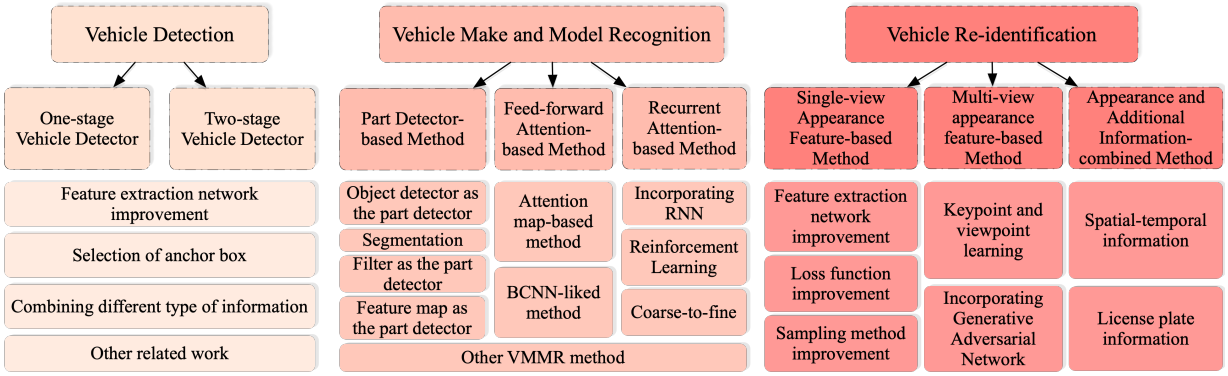


Figure 2.4: The classification of deep learning-based methods for each task.

2.3 Vehicle Detection

Vision-based VD mainly focuses on detecting the vehicles from an image and recognizing the type of the vehicle. VD method can be widely used in autonomous driving, toll collection, traffic surveillance, and security-related area (e.g., gate monitoring, suspicious vehicle searching) [234].

Traditional VD methods rely on extracting handcrafted features such as SIFT, SURF, HOG, LBP, Haar-like, and optical flow. The introduction and related applications of these handcrafted features can be found in References [234, 82, 192, 193, 190, 191, 188]. Recently, deep learning-based methods have shown excellent performance in VD. The deep learning-based VD method is usually developed based on the general object detection method. Therefore, we will first briefly review the general object detection methods and then introduce the modification of these methods.

2.3.1 Object Detection Method

Recently, researchers have proposed a significant number of deep learning-based object detection methods. These methods significantly advance the object detection precision, and can be classified into two categories: two-stage detector (such as R-CNN [124], Fast-RCNN [125] and Faster-RCNN [216]), and one-stage detector (such as YOLO [213], SSD [173], CornerNet [160], and CenterNet [111]).

Two-stage detectors

Two-stage detectors need two steps (i.e., generating region proposals and classifying these proposals) to get the detection results. R-CNN [124] is one of the earliest deep learning models that was proposed for the task of object detection. R-CNN uses the Selective Search [251] method to generate a set of region proposals. Then, these proposals are wrapped to the same size and sent to CNN to extract features. The Support Vector Machine (SVM) [96] is used to classify these extracted features. Finally, the Non-Maximum Suppression

(NMS) method is used to filter these proposals for each class independently. However, R-CNN has several drawbacks. First, the feature extraction network and the classifier of R-CNN need to be trained separately, and the training takes a long time. Second, this model cannot be used for real-time detection as it takes 47 seconds to process one image [125].

To address the issues mentioned above, Girshick et al. [125] proposed Fast-RCNN. Instead of feeding the region proposals to CNN, Fast-RCNN directly processes the original image and obtains the feature maps. Then, the generated region proposals are projected to the feature maps. A Region of Interest (ROI) pooling layer is introduced to reshape the selected feature maps to a fixed size so that they can be classified by the FC layers.

Both R-CNN and Fast-RCNN adopts the region proposal methods that cannot be trained to generate better region proposals. In addition, their region proposal step significantly slows down the whole network [216]. Faster-RCNN [216] uses Region Proposal Network (RPN) to generate the region proposals. RPN uses a small window sliding over the feature maps. At each feature map location, it simultaneously generates multiple region proposals. RPN improves the detection accuracy and speed of the detector. RPN also allows Faster-RCNN to be trained end-to-end. These two-stage detectors usually have high detection accuracy, but the detection speed is slow and cannot achieve real-time detection.

One-stage detectors

To achieve real-time object detection, one-stage detectors are proposed. One-stage detectors directly predict the bounding box parameters for the objects in the image and classify these objects at the same time. One-stage detectors often have high detection speed but relatively low detection precision.

YOLO [213] is a popular one-stage detector. YOLO treats the image as a S $\times$ S grid. Each grid cell can generate B bounding boxes, and each box predicts one object. For each bounding box, YOLO predicts the center offset (x, y) related to the corresponding grid cell, the width and height of the bounding box, and the box confidence score. The confidence score represents whether the cell contains an object. In addition, k class probabilities are predicted for each grid cell. The final classification scores for the bounding box are obtained by multiplying confidence scores and the class probabilities. YOLO also applies NMS to remove duplicate predictions, which increases the mean Average Precision (mAP) by 2-3%. YOLO can be trained end-to-end, and it can achieve a detection speed of 45 Frames Per Second (FPS) on an NVIDIA Titan X GPU with mAP 63.4% on VOC2007 dataset [117]. YOLO has several drawbacks. First, in YOLO, each grid cell is designed only to predict a small number of boxes, which limits YOLO's ability when it comes to detecting multiple closed objects. Second, YOLO has relatively low recall compared to two-stage detectors. Furthermore, YOLO makes significant localization errors [214].

YOLOv2 [214] is an extended version of YOLO. The basic structure of the model is called Darknet-19. YOLOv2 adopts Batch Normalization [146], which can accelerate the convergence speed and increase mAP by 2%. Instead of using the FC layer to predict the bounding box parameters, YOLOv2 predicts the offsets and the confidence score for five

default anchor boxes at each feature map cell. The authors utilized k-means to cluster the ground-truth bounding box of the objects in the training set to select the size of default anchor boxes for each feature map cell.

YOLOv3 [215] is a fine-tuned version of YOLOv2. YOLOv3 adopted Darknet-53 as the feature extraction network. YOLOv3 predicts bounding boxes from three feature maps of different scales. In some layers, the feature maps were upsampled and concatenated with the feature maps generated by earlier layers. The combination allows the model to extract more fine-grained information.

SSD [173] is another popular one-stage detector. The feature extraction network of SSD is the truncated VGG-16 [233]. Instead of randomly generating bounding box parameters, SSD directly predicts the classification results along with the bounding box adjustments for each default anchor box at each feature map cell. The size of the default anchor boxes is manually defined and has different scales and aspect ratios. SSD generates predictions of different scales from different feature maps with different resolutions. The authors' idea is to use lower-level layers to predict small objects and higher-level layers to predict large objects.

YOLOv2, YOLOv3 and SSD all use the anchor box. The use of default anchor boxes significantly improves the accuracy of these models. However, there are two major drawbacks: (1) There are many anchor boxes are predicted (e.g., SSD: more than 8K) and only a small part of them are matched with the ground truth boxes, causing an imbalance in the number of the positive samples and negative samples. Thus, the training efficiency is severely affected. (2) A lot of hyperparameters are introduced, such as the size and the number of default anchor boxes. Researchers need to spend a lot of effort to select these hyperparameters.

Recently, a new type of one-stage detector for predicting the bounding box has been proposed. CornerNet [160] does not use the default anchor box. CornerNet conducts the detection by predicting a pair of keypoints of the object (the top-left keypoint and the bottom-right keypoint). The feature extraction model is the hourglass network [202], which has a great ability to generate keypoints of the object. In CornerNet, a novel pooling layer, corner pooling, is introduced to locate the keypoints of the bounding box.

CenterNet [111] uses three keypoints to locate the object. Each object is represented by a center keypoint and a pair of corners. The authors proposed the cascade corner pooling to enrich the information collected by the pair of corners. Moreover, the center pooling is proposed to collect recognizable information in the central regions. Even though CornerNet and CenterNet are the one-stage detectors, their detection accuracy is higher than that of the general two-stage detectors. As these two models use the keypoint instead of the anchor box to locate the object, they have excellent performance on detecting occluded objects. The use of keypoints also saving the work of finding a set of default anchor boxes. However, one of the drawbacks of these two models is they have a relatively low detection speed. The speed of processing an image of CornerNet511-104 is 300 ms and CentnerNet511-104 is 340 ms [111] (on an NVIDIA Tesla P100 GPU).

These two types of object detection methods have a powerful ability to detect general objects. The characteristics of these methods are summarized in Table 2.2. Researchers

Table 2.2: Summary of Commonly Adopted Object Detection Methods

Type	Method	Backbone	Characteristics
Two-stage	R-CNN (2014) [124]	AlexNet	Selective Search, features are extracted from each region proposal
	Fast-RCNN (2015) [125]	VGG	Selective Search, ROI pooling layer, proposal projection, bounding box regression
	Faster-RCNN (2015) [216]	VGG	RPN, anchor box, trained end-to-end
One-stage	SSD (2016) [173]	VGG	Multi-layer bounding box prediction, default anchor box
	YOLO (2016) [213]	GoogLeNet	Dividing the feature map into grid and directly predicting bounding box parameter
	YOLOv2 (2017) [214]	Darknet-19	Default anchor box, multi-scale training, predicting on each feature map cell
	YOLOv3 (2018) [215]	Darknet-53	Feature extraction improvement, feature maps concatenation
	CornerNet (2018) [160]	Hourglass	Pair keypoints detection, corner pooling
	CenterNet (2019) [111]	Hourglass	Three keypoints prediction, center pooling

have made numerous modifications to improve their vehicle detection performance. In the following section, we will introduce the VD method.

2.3.2 Vehicle Detection Method

Vehicle images are usually captured by the onboard camera or the surveillance camera. Vehicles in the image can vary significantly in size, and the background is complex and changeable. Compared with the general object category, the number of vehicle types is relatively small (around 10 in KITTI [122] and CompCars [265] datasets). Based on these features, the general object detection method is modified accordingly and adopted for VD. The modification for the general object detection methods mainly focuses on changing the feature extraction network and designing a better anchor box selection method.

Feature extraction network improvement

A common approach to improve the model's detection accuracy and processing speed is changing the feature extraction network or modifying the two-stage detector's region proposal method. The authors in Reference [89] observed that the region proposal methods [124, 125, 216] generate proposals based only on the feature maps of the same scale, which could cause a mismatch for objects with large scale variations in an image. Therefore, they used multiple proposal networks to generate object proposals from multiple feature maps (different resolutions) to detect the objects of different scales. As a result, the model could generate high-quality proposals.

To take advantage of the smaller receptive region of the lower layers, Gao et al. [121] combined the multi-layer proposal method (concatenate features from different layers), multi-scale proposal method (using multiple convolutional kernels like the Inception module), and Residual block embedding method (residual module) to improve Faster-RCNN

for VD. Sang et al. [227] removed the redundant convolutional layers in higher layers of YOLOv2 by considering that only six vehicle types need to be distinguished in their dataset. They also adopted the multi-layer feature fusion strategy to enhance the feature extraction ability of the network.

Hao et al. [132] proposed a Multi-Target Corner Pooling (MTCP) to locate corners of the vehicles. A corner matching method is introduced to complement corners when the vehicles are blocked. Therefore, the proposed vehicle detection model can achieve excellent performance in complex traffic scenes (e.g., when vehicles are occluded).

Selection of anchor box

The selection of default anchor boxes is also essential to the vehicle detector. Gao et al. [121] used smaller default anchor boxes to improve the vehicle detection ability of Faster-RCNN. Sang et al. [227] adopted the k-means++ clustering [27] algorithm to cluster the bounding boxes of the training images to select the size of the default anchor boxes. Wu et al. [260] observed that the size of the default boxes selected by the k-means++ algorithm can help YOLOv2 to detect the vehicles with the normal size (e.g., sedan), while the generated boxes are usually not suitable for the vehicles with unusual size (e.g., minivan). Therefore, the authors extended [227] and proposed Rk-means++ to enhance the model adaptability to vehicles of various sizes to achieve multi-scale detection. Rk-means++ applies a certain proportion to the two ratios selected by the k-means++ algorithm to generate anchor boxes that match vehicles with different sizes. Thus, the localization ability of YOLOv2 is improved with the help of the Rk-means++.

Combining different type of information

Another way to improve VD model performance is to combine the vehicle-related information obtained by different sensors such as radar and camera. He et al. [137] introduced a coordinate transformation method to transform the radar information from the radar coordinate system to the image coordinate system. The combination of radar information and image information improved the model stability in bad weather conditions. Instead of extracting the 2D information of the vehicle, Li et al. [162] proposed a video-based vehicle detector. This detector uses the 3D-CNN to capture the temporal information from video frames.

Other related work

In addition to the on-road image VD method, another related work is the VD in aerial (satellite) image. The aerial image is taken by the cameras mounted on the Unmanned Aerial Vehicle (UAV) or the satellite. The aerial image is usually captured at the height of 100 to 1,000 m (sometimes even higher) [107]. The aerial image has several features: (1) The size of the vehicles is very small (vehicles only take up 30 $\times$12 pixels in an aerial image with a resolution of 5,616 $\times$ 3,744 pixels [171]); (2) vehicles have various orientations; (3)

vehicles are usually occluded by objects (e.g., tree); (4) the background is intricate and some objects can easily cause the false alarm. These four characteristics are the main challenges of aerial image-based VD methods.

Jiang et al. [151] adopted a graph-based superpixel segmentation method [119] to segment the image into a set of superpixels, and then used a deep CNN to determine whether the image patch (superpixel) contained a vehicle. The superpixel segmentation method could reduce the number of sliding windows used, thereby accelerating the processing speed. Deng et al. [107] proposed one network to extract the vehicle-like regions on combined hierarchical feature maps and another network to predict the orientation and type of a vehicle. As the original aerial image (5616×3744 pixels) is very large, the authors proposed to crop the original image into 48 image blocks. These image blocks are stitched together after getting the prediction results.

2.3.3 Summary of Vehicle Detection Method

The two-stage object detectors usually have a high detection accuracy but a relatively low detection speed. On the contrary, the one-stage object detectors typically have a very fast detection speed, but the accuracy is relatively low. Aiming at the characteristic of these two types of object detectors and the features of vehicles, many studies have proposed methods to improve the model's accuracy on detecting vehicles while maintaining the detection speed. The general way of modification can be summarized as follows: (1) Changing the size of the image. (2) Choosing the size of the default anchor box. (3) Utilizing features of different scales. (4) Removing redundant layers. Table 2.3 summarizes the backbone, model type, image type, and characteristics of the vehicle detectors.

As we mentioned before, CornerNet and CenterNet use the keypoint to locate the object, and these two models have a higher detection accuracy than the general two-stage detectors. Especially, they have a stable performance on detecting occluded objects. Therefore, we believe using the keypoint is the future direction to improve the performance of vehicle detectors.

2.4 Vehicle Make and Model Recognition

After locating vehicles in the image using the VD method we mentioned above, vehicles are cropped out from the original image and then sent to the VMMR module for finer recognition. In this section, we will review the deep learning-based method that is aimed at addressing the VMMR task. Various deep learning-based methods have been proposed to solve the VMMR problem, such as the part detector-based method, recurrent attention-based method, and feed-forward attention-based method. In addition, VMMR is the subcategory of fine-grained recognition, therefore, some related methods are also introduced. In the following, we will introduce these models comprehensively.

Table 2.3: Summary of Vehicle Detection Methods

Model Type	Method	Image Type	Backbone	Characteristics	Comments
Two-stage	MS-DCNN (2016) [89]	On-road	Faster-RCNN	Multi-scale region proposal	The model generates high quality region proposals
	Gao et al. (2017) [121]	On-road	Faster-RCNN	Using small scale default anchor box, feature maps concatenation	Excellent performance on detecting small vehicles
	Jiang et al. (2015) [151]	Aerial	-	Superpixel segmentation method, sliding window region proposal	Superpixel segmentation method can accelerate the detection speed
	Deng et al. (2017) [107]	Aerial	Faster-RCNN	Region proposal on combined hierarchical feature maps	High computational cost and low detection speed
One-stage	Sang et al. (2018) [227]	On-road	YOLOv2	K-means++ default anchor box selection, multi-layer feature fusion, loss refine	Strong weather adaptability (daytime and night)
	He et al. (2018) [137]	On-road	YOLOv2	Coordinate transforming method, radar and image information fusing	Stable performance in bad weather conditions
	3D-DETNet (2018) [162]	On-road	YOLOv2	Combining temporal information, focal loss	Better performance than YOLO on rainy and night scenarios
	Hao et al. (2019) [132]	On-road	CornerNet	Multi-target corner pooling layer, corner matching	Excellent performance when vehicles are occluded

2.4.1 Part Detector-based Method

VMMR can be considered as a special scenario of image classification. VMMR requires the model to have the ability to extract subtle features of a vehicle. Part detector-based methods address the VMMR problem by using different strategies (e.g., adopting an object detector or grouping filter channels as the part detector) to locate the discriminative parts (e.g., headlight, wheel) of a vehicle. Then, the selected part regions are sent to the feature extraction network to extract the features of these parts. Finally, part features and global features are processed by the classification network to generate the classification probabilities for each category. Generally, the final probability of each category is the average of all the classification results. The vehicle model of the input image is the category with the highest probability. The detailed information of these part detector-based methods is introduced below.

Object detector as the part detector

A Part-based R-CNN was proposed in Reference [278]. In their work, R-CNN was adopted to detect an object and localize its semantic parts (e.g., bird head, dog nose). The object and its parts were initially considered as independent object categories. A one-versus all linear SVM was trained based on the features of the region proposals to score each region. More specifically, each candidate detection region is assigned a score of $w_i^T f(x)$, where w_i is the learned SVM weight for region x, $f(x)$ is the feature (extracted by CNN) of the candidate region. A region with an overlap $\geq$0.7 with any ground-truth bounding box (object or part) is taken as a positive for that object or part, and a negative region has an overlap $\leq$0.3 with any ground-truth box. Then, one object detector and several part detectors are trained with the help of object bounding box annotations and semantic part annotations. In addition, geometric constraint functions were proposed to constrain each part detector to detect the corresponding part. Similarly, Huang et al. [143] also adopted R-CNN as the part detector. In their model, they trained one global vehicle detector and seven part detectors (i.e., left and right turn signal lamps, two front lights, two rear-view mirrors, and one air-inlet grille). Liu et al. [178] trained a part localizer and a multi-label predictor for each part with the part attribute descriptions (e.g., wing color gray of a bird).

For these methods [278, 143, 178], they have an excellent performance in localizing parts. However, the part bounding box annotation or the part attribute descriptions need to be provided to train the model, which may limit the application of these methods, since not all datasets provide these detailed annotations. Besides, the human-defined part labels may introduce noise and prevent the model from achieving optimal performance. Therefore, some part detector-based methods were proposed, which do not require the part annotations to train the model.

Segmentation

According to the observation that fine-grained categories (e.g., different vehicle models within the same type) share similar shapes, Krause et al. [156] found the fine-grained

objects can be aligned via segmentation (partitioning image into different parts at the pixel level), and a figure-ground segmentation was sufficient to determine the pose of the object and locate its parts. Therefore, the authors extended the co-segmentation [127] method to segment the parts of the object, and aligned pairs of images with similar poses. Then, the authors sampled points across all images based on these alignments. A fixed area (corresponding to the bounding box of the object) around each point was taken as a part region. Although the authors adopted R-CNN as the object detector to detect the object during testing, their model does not require part annotation.

Filter as the part detector

Alternatively, some methods [261, 281] group the filter channels as the part detector. A two-level attention model was proposed in Reference [261]. For object-level attention, the authors trained a CNN (FilterNet) to predict the general object category (e.g., cat, vehicle, dog) of an image. They adopted Selective Search to generate multi-scale and multi-view region proposals. Then, these proposals were sent to FilterNet to filter out the proposals (according to the generated object score) that not relevant to the object. Then, the selected patches were used to train another CNN (DomainNet). By calculating the cosine similarity of the filters in the Conv4 layer of DomainNet (AlexNet), filters were clustered into groups and served as part detectors (part-level attention). Each group of filters pays more attention to a particular part (e.g., one group of filters focuses on the headlight, and another focuses on the logo). Similarly, Zhang et al. [281] found that some filter channels could respond to specific patterns consistently (e.g., one channel always has a high response to a particular part). Therefore, they proposed a two-step picking strategy to pick up some distinctive filter channels as a set of part detectors. Wang et al. [255] proposed a Discriminative Filter Learning (DFL) module. In the DFL module, the 1×1 filters were taken as the patch detectors to generate patch response maps. The authors introduced a Cross-Channel Pooling layer to supervise the patch detectors. The filter supervision significantly improves model performance. These models [261, 281, 255] do not require the use of the object-level bounding box.

Feature map as the part detector

The feature map can also indicate the location of the part. Cross-domain hallucination network was proposed in Reference [165]. The authors proposed to use the hallucination network to reconstruct the low-resolution surveillance images under the guidance of the high-resolution web images (the same vehicle models). Then, some channels of the feature maps with maximum responses to some parts are manually selected. The part regions of the vehicle are cropped from the original image according to the selected locations of the cells in the feature map. However, manually selecting the feature map is time-consuming and error-prone. More efficient ways of using the feature map will be introduced in the next sections.

Table 2.4: Overview of Representative Part Detector-based Recognition Methods. OP: object detector as the part detector, SEG: segmentation, FP: filter as the part detector, MP: feature map as the part detector

Method	Backbone	Characteristics	Comments
Zhang et al. (2014) [278]	CaffeNet	OP, part detector (R-CNN), geometric constraint function	The geometric constraint function significantly helps the model localize the object and its parts
Krause et al. (2015) [156]	VGG	SEG, object detector (R-CNN), sampling points from alignments	Requires bounding box to refine segments; the model accurately extracts the foreground
Xiao et al. (2015) [261]	AlexNet	FP, selective search, filter cluster according to cosine similarity	Low recognition accuracy; part localization without using the bounding box
Zhang et al. (2016) [281]	VGG	FP, selective search, picking filter channels as part detectors based on the response scores	High small objects recognition accuracy; part localization without using the bounding box
Huang et al. (2016) [143]	AlexNet	OP, one global vehicle detector and seven part detectors (R-CNN)	Only tested on a small dataset; low recognition accuracy
DFL-CNN (2018) [255]	VGG	FP, patch response maps, patch filter supervision	High accuracy on recognizing objects of different sizes; simple and efficient; end-to-end training
Lin et al. (2018) [165]	CaffeNet	MP, image reconstruction, part hallucination network	Mannully selecting the feature map as the part detector is time-consuming and error-prone

Summary

In general, the ways of part detector-based models detect the part regions are: (1) using the object detector; (2) sampling points from alignments; (3) grouping the filter channel; (4) directly cropping the region with high activation values in the feature map. These models often require multi-step training. In Table 2.4, we summarize the features of the part detector-based recognition methods. We noticed that part detector-based models still have relatively low recognition accuracy even though they adopted some great methods to locate an object's discriminative parts. By analyzing these models, we believe that these models' accuracy can be further improved by choosing a more powerful standard CNN architecture (e.g., ResNet, DenseNet) as the backbone.

2.4.2 Feed-forward Attention-based Method

Incorporating the feed-forward attention mechanism is another way to enhance the standard CNN architectures. These models expand the CNN architectures by generating the attention masks based on the generated feature maps. These attention masks are then used to attend the feature maps to select the discriminative regions of an image. In addition, generating the bilinear feature to capture the pair-wise feature interaction can also improve the model performance.

Attention map-based methods

Zagoruyko et al. [273] proposed activation-based and gradient-based attention transfer methods to train a teacher-student network. More specifically, for the activation-based attention transfer method, the authors introduced three attention mapping functions to generate attention maps:

$$F_{sum}(Z) = \sum_{i=1}^{C} |Z_i|, \;\; F^p_{sum}(Z) = \sum_{i=1}^{C} |Z_i|^p, \;\; F^p_{max}(Z) = max_{i=1,..,C}|Z_i|^p, \tag{2.11}$$

where $Z \in \mathbb{R}^{H \times W \times C}$ represents the feature maps, H and W denote the height and width of the feature maps, C is the channel of the feature maps, $Z_i = Z(:,:,i)$, $p > 1$. By observing the attention map, the authors found that in the lower layer, the activation value of the attention map is high for some points, in the middle layer, the activation value is high for some regions such as a human's mouth or a vehicle's lights, and in the top layers, the attention map focuses on a larger area. (e.g., the human face). For the gradient-based attention, the authors defined the attention as the gradient with regard to the input. Both attention transfer methods can improve the performance of the student network, and activation-based attention works better. Spatially Weighted Pooling (SWP) [141] directly predefines a set of spatially weighted masks and takes them as the attention maps to pool the image features.

Multi-Attention CNN (MA-CNN) was introduced in [287]. In this model, a channel grouping layer was proposed to the cluster channels of the feature maps, and these channels respond to spatially correlated patterns. The attention map is the weighted sum of these channels. More specifically, the authors defined N FC layers to regress the channels of feature maps into N groups to detect N parts. Each FC layer takes the feature maps of the image as input and produces a vector $V_i(Z) = [v_1, ..., v_c]$ over different C channels. The scalar v_j indicates whether the channel j belongs to part i. A part attention map $M_i(Z)$ is then defined as follows:

$$M_i(Z) = \sigma(\sum_{j=1}^{C} v_j * [Z]_j), \tag{2.12}$$

where Z represents the feature maps, $[Z]_j$ denotes the j^{th} channel of Z, and σ denotes the sigmoid function. The attention map $M_i(Z)$ is then used to pool Z to generate the part features:

$$P_i(Z) = \sum_{j=1}^{C}([Z]_j \odot M_i(Z)), \tag{2.13}$$

where $\odot$ denotes the element-wise product. Finally, the part features and the global object feature are concatenated together and classified by the FC layer.

Instead of localizing the larger parts from the original image and then localizing the smaller parts from the larger parts as Recurrent Attention CNN (RA-CNN) [120], Multi-part Convolutional Attention Network (M-CAN) [290] simultaneously learns discriminative object parts from feature maps of different scales. To remove the background noise, the

authors used the Class Activation Map [292] as the object detector to select the object features. Then, the selected object features of different scales were sent to the part localization network. In the part localization network, the deconvolutional layer [275] was adopted to generate a single-channel attention map. The highest value in the generated attention map was taken as the part region. Rodriguez et al. [222] proposed a modular attention architecture. The attention map is generated by a convolutional layer and normalized by a spatial softmax function. The module can be added to any layer of the current existing CNN architectures, so it is a flexible architecture and can be trained in parallel to the original network. Jetley et al. [148] defined a compatibility function, which combines the weighted local feature and global feature, forcing the model to focus on the salient region of the image and suppress the background information.

BCNN-liked methods

Bilinear CNN (BCNN) was proposed in Reference [166]. In this model, two CNNs were utilized to generate two distinctive feature representations. The authors' motivation was to use one network to locate discriminative parts, and another to extract the features of these part regions. These two representations were combined together through an outer product, obtaining a global image representation (bilinear vector). Let $Z_a \in \mathbb{R}^{H\times W\times C_a}$ and $Z_b \in \mathbb{R}^{H\times W\times C_b}$ denote the feature maps generated by two CNNs. The bilinear feature at location l is defined as follows:

$$B(l, Z_a, Z_b) = Z_a(l)^T Z_b(l), \tag{2.14}$$

where $Z_a(l) \in \mathbb{R}^{1\times C_a}$, $B \in \mathbb{R}^{H\times W\times C_a C_b}$. Then, the bilinear feature is processed either by summing or max pooling along the spatial location, obtaining the bilinear vector. The authors also showed the relationship between the bilinear model and various orderless texture descriptors such as Bag-of-Visual-Words model [102].

As the generated bilinear feature is a high-dimensional image representation, the computational cost is very high. There are many methods [155, 270, 246] proposed to improve BCNN. Kong et al. [155] introduced a low-rank bilinear classifier to approximate the high-dimensional bilinear feature, which significantly reduced the computational complexity. The size of the proposed model is 0.8 MB, while the original BCNN is 200 MB [155], and they have a similar recognition accuracy. BCNN only utilized the feature representation generated by the last convolutional layer. This feature representation is insufficient to capture all discriminative part features [270]. In Reference [270], a Cross-layer Bilinear Pooling (CBP) method is proposed to obtain the inter-layer part feature interaction. The authors cascaded multiple CBP modules and constructed a Hierarchical Bilinear Pooling (HBP) framework to capture multiple inter-layer feature interactions. HBP uses all the information of the feature maps. The information can contain noisy features, which may degrade the final recognition accuracy [246]. Tan et al. [246] introduced an Aggregated Slack Mask model to selectively extract salient features from multiple layers, thereby making up for the drawback of the HBP framework. The slack mask is a binary image mask generated based on the feature maps.

Table 2.5: Overview of Representative Feed-forward Attention-based Methods. AM: attention map-based method, BL: BCNN-liked method

Method	Backbone	Characteristics	Comments
BCNN (2015) [166]	VGG	BL, feature maps combination through outer product	High computational cost; provided a way of extracting discriminative features
Zagoruyko et al. (2016) [273]	WRN [272], ResNet	AM, teacher-student network, attention transfer study	Explored the working mechanism of the attention map; results show activation-based attention works better
MA-CNN (2017) [287]	VGG	AM, channel grouping layer, channel grouping loss	Accurate multi-part localization without using bounding box annotation
Kong et al. (2017) [155]	VGG	BL, low-rank bilinear classifier	Very small model size; comparable recognition accuracy with BCNN
Jetley et al. (2018) [148]	VGG	AM, compatibility function combine local and global features	Shows the effectiveness of attention in multiple applications; the model is robust to adversarial attacks
Rodriguez et al. (2018) [222]	WRN	AM, modular attention architecture	Small model size, fast recognition speed
M-CAN (2018) [290]	VGG	AM, simultaneously extract multi-scale part features	Fast attention process; relatively low recognition accuracy
Tan et al. (2019) [246]	ResNet	BL, selectively extracting features with the slack mask	High accuracy on recognizing large objects

Summary

In this section, we introduced different methods for generating and using the attention map to select discriminative features. Moreover, the mechanism and the improvement of BCNN were introduced accordingly. Feed-forward attention-based methods usually can be trained end-to-end and have a high recognition speed. The features of these methods are shown in Table 2.5.

The BCNN-liked models often directly use the feature maps without feature selection. We think BCNN-liked models can be further improved by using the attention map to select the discriminative features.

2.4.3 Recurrent Attention-based Method

The recurrent attention-based method is inspired by the observation that humans tend to focus their attention on parts of a scene within multiple glimpses because of their built-in visual and biological attention mechanisms [108]. Inspired by the biological attention mechanisms, Recurrent Neural Networks (RNNs), such as Long-Short-Term Memory (LSTM) [139], are introduced to combine with the CNNs, which allow models [284, 259] to pay attention to different discriminative parts of the object. Similarly, some methods [179, 285] adopt the Reinforcement Learning (RL) algorithm to train a network to selectively find the semantic region of an object without using the part annotations. In addition, some methods [120, 249] adopt a coarse-to-fine strategy to recognize the object.

Incorporating RNN

Diversified Visual Attention Network (DVAN) [284] combines LSTM to extract the features of semantic parts through multiple timesteps. In DVAN, a window is utilized to slide over the original image to generate diversified attention canvases (region proposals). These attention canvases were resized to the same size, and then sequentially processed by VGG to extract features. The generated feature maps were sent to the proposed Visual Attention Model (VAM) to generate attentive features for different parts. VAM consists of the LSTM. Different parts are found through multiple timesteps. During each timestep, VAM generates an attention map that focuses on a region of the feature maps. The attention map is generated by integrating the hidden state of LSTM and feature maps Z_t:

$$l_{t,i} = \frac{exp(W_{h,i}^T h_{t-1} + W_{x,i}^T Z_t)}{\sum_{j=1}^{K^2} exp(W_{h,j}^T h_{t-1} + W_{x,j}^T Z_t)}, \tag{2.15}$$

where $l_{t,i}$ is the value of i^{th} location of the attention map l at time t, h_t is the hidden state of LSTM. The attention map can focus on K^2 location of Z_t. The attentive feature z_t is the input of LSTM and it is generated by using the attention map to pool Z_t: $z_t = \sum_{i=i}^{K^2} l_{t,i} Z_{t,i}$. The hidden state h_t is also used as the feature for classification. This model can locate different prominent regions of the object. However, DVAN has a relatively low recognition accuracy compared to other recurrent attention-based models, and the whole model needs to be trained separately.

In BCNN [166], the pooled bilinear features are summed along the spatial location, which ignores the location information of the features. To preserve the location information, Wu et al. [259] introduced a Spatial LSTM to generate spatially meaningful hidden representations. These hidden representations can capture the spatial relationship of part features.

Reinforcement Learning

Fully Convolutional Attention Network (FCAN) was proposed in Reference [179]. The attention procedure was formulated into a Markov Decision Process. The authors reused the same feature maps across all time steps. At each timestep, the model samples an attention location l_t according to the attention selection policy. The RL algorithm was used to optimize the attention network to focus on discriminative regions. The reward strategy is: at the first timestep, the attention network can receive a reward if the image is classified correctly; at other time steps, a reward is given when the image is correctly classified and the loss decreases. Otherwise, there is no reward. The model achieves a high recognition accuracy, but it requires multi-step training. Zhao et al. [285] proposed a visual attention module. The module constructs the attention map matrix according to a selected attention location, and the matrix is used to transform a raw image into a focused image (i.e., one part clear and others fuzzy). To find a good attention location, the authors adopted the Q-learning algorithm to find the key areas of the image.

Coarse-to-fine

Fang et al. [118] proposed a coarse-to-fine strategy, which iteratively generates finer parts according to the heatmap. The heatmap is an average of the channels of the feature maps of all training images. The obviously isolated tight rectangular blocks in the heatmap were selected as the discriminative region. Fu et al. [120] proposed a RA-CNN, which iteratively extracts the finer discriminative region and produces classification results for each scale. The general structure of RA-CNN consists of the feature extraction network, Attention Proposal Network (APN) and classification network, and the general structure is stacked twice. In APN, a FC layer is used to generate the part region: $[i_x, i_y, i_l] = f(Z)$, where Z represents the feature maps, $f(\cdot)$ represents the FC layers, i_x, i_y and i_l present the center coordinates and length of the predicted square region i. The region is then represented by a top-left (tl) point and a bottom-right (br) point. Then, an attention map is generated based on the location information. An element-wise multiplication is performed between the input image and the attention mask to crop the selected region X^{att} from the input image X:

$$X^{att} = X \odot M(i_x, i_y, i_l) \tag{2.16}$$

$$M(\cdot) = [h(x - i_{x(tl)}) - h(x - i_{x(br)})] \cdot [h(y - i_{y(tl)}) - h(y - i_{y(br)})] \tag{2.17}$$

where $h(\cdot) = 1/(1 + exp^{-kx})$. When k is large, $M(\cdot)$ works as the two-dimensional box-car function (attention map) where the value of the selected regions is one, others are zero. $M(\cdot)$ can be used to approximate the cropping operation and the parameters can be optimized through backpropagation. To get a finer part region, X^{att} is amplified to the same size as the input image with the bilinear interpolation function and took as the input of the next feature extraction network. The model has excellent performance, and more importantly, the training of RA-CNN only needs category labels.

Similarly, Simonelli et al. [232] constructed a model, which is an ensemble of n same structure networks, to sequentially process and zoom the image. The zoomed image is produced by a proposed focus module. The module contains a binary mask that is generated based on the weights of a linear classifier. Tian et al. [249] proposed a Selective Multi-Convolutional Region (SMCR) feature extraction method. This method extracts coarse local feature regions by comparing the activation value in the heatmap with the mean activation value. The heatmap is generated by summing the feature maps along the channel direction. The author introduces an attention module similar to APN, which extracts finer part features in an iterative manner. Zheng et al. [286] proposed a progressive-attention CNN by extending the MA-CNN.

The aforementioned models [120, 232, 249, 286] can achieve high recognition accuracy, but they have a large model size and require fine-tune the submodels separately (multi-step training). Liu et al. [169] proposed a Bidirectional Attention-Recognition Model. In this model, an attention agent (RPN) was used to propose part regions, and a recognition agent was used to extract the region features. The recognition result from the recognition agent is transferred to the attention agent through a feedback path so that the two agents can reinforce each other. Thus, the model can be trained end-to-end. In addition, the authors proposed a Multiple Random Erasing data augmentation method. This method randomly

Table 2.6: Overview of Representative Recurrent Attention-based Methods. IRNN: incorporating RNN, RL: reinforcement learning, CTF: coarse-to-fine

Method	Backbone	Characteristics	Comments
FCAN (2016) [179]	ResNet	RL, reusing the same feature maps	High recognition accuracy; the model can effectively locate different part regions
DVAN (2017) [284]	VGG, LSTM	IRNN, attention canvas generation, diversified attention regions	low recognition accuracy; the attention canvas generation is a fixed process, and a better canvas cannot be obtained through training
RA-CNN (2017) [120]	VGG	CTF, stacking CNNs to extract features of different scales	large model size; high part localization accuracy
Zhao et al. (2017) [285]	VGG	RL, focused image, Q-learning	Only tested on small scale datasets; unclear model performance when vehicle pose changes
ID-CNN (2018) [249]	VGG	CTF, SMCR feature extraction, iteratively part cropping	Stable performance when vehicle pose changes
Wu et al. (2019) [259]	VGG, LSTM	IRNN, Bilinear pooling, spatial recursive encoding with attention	Excellent performance on recognizing small objects; LSTM can help the model keep the spatial relationship of different parts
Zheng et al. (2020) [286]	VGG	CTF, Part features are extracted progressively	Accurate finer part localization ability; long training time; many hyperparameters

selects multiple rectangle areas of the training image and erases their pixels with random values. Ma et al. [185] proposed a Recurrent Attention Unit to recurrently extract and combine discriminative features of different scales.

Summary

The recurrent attention-based method can generate an attention location based on the past attention location, which enables the model to focus on different discriminative regions of an object. The features of these recurrent attention-based methods are shown in Table 2.6. This type of method achieves high recognition accuracy by accurately locating different or finer discriminate regions of the object. However, one weakness of this type of method is that the submodules of these models usually need to be fine-tuned separately to achieve optimal recognition results.

For the models of coarse-to-fine subcategories, they either stack networks of the same structure multiple times or consist of multiple subnetworks, which makes them have a large model size. For future work, more efforts can be made to simplify the models to allow them to work in a portable device and have a faster recognition speed.

2.4.4 Other VMMR Methods

There are some methods that do not belong to any of the categories we mentioned above, but they provide potential directions to improve the model. Xie et al. [263] collected super-type hyper class images (coarse class labels, e.g., car, cat) and factor-type hyper class images (different view of fine-grained class labels, e.g., front view of Audi Q5) as the auxiliary datasets to augment model training. Sochor et al. [236] utilized the 3D bounding box annotation. These annotations allow each side of the vehicle (e.g., roof, front, rear) to be identified. Relying on the 3D bounding box annotation, the image is unpacked into a plane, and the sides are encoded as three 2D vectors.

In Reference [280], the classification and similarity constraint of the fine-grained representations are jointly optimized. Instead of learning one image feature representation once a time, the network took four images as the input of a CNN to generate four image feature representations. The authors developed the triplet loss (will introduce in Section 2.5.1) to a quadruplet loss $L(a, x^+, x^-n, m_1, m_2)$:

$$\begin{aligned} L = & \frac{1}{2N}\sum_{i=1}^{N} max\{0, D(a_i, x^+) - D(a_i, x^-) + m_1 - m_2\} \\ & + \frac{1}{2N}\sum_{i=1}^{N} max\{0, D(a_i, x^-) - D(a_i, n_i) + m_2\}, \end{aligned} \tag{2.18}$$

where x^+ is the image of the same fine-grained class as a (the same vehicle make and model), x^- is the image of the same coarse class as a (the same vehicle make), n is the negative sample (different vehicle make and model), $D(x, y)$ is the distance function, and m_1 and m_2 are the margins. The quadruple loss can be seen as a combination of two triplet losses. The negative sample set in the triplet is further classified into two subsets.

FC layer is often employed to produce the probability for each class. In the original Cross-Entropy Loss (CEL) function, only the probability related to the ground-truth class is optimized in a training batch, the probability of other classes is ignored. In Reference [163], a Dual CEL (DCEL) was proposed to constrain the increase of non-ground-truth class probabilities. Chen et al. [92] outlined a progressive transfer learning problem and proposed a hierarchical transfer learning approach for fine-grained recognition. In their work, multiple Generative Adversarial Networks (GANs) [126] (generators for different classes and discriminators with different scales) are used to learn transferable information. Ma et al. [186] proposed a Channel Max Pooling (CMP) layer. CMP conducts the max pooling operation along the channel direction of the feature maps. CMP layer gathers the significant features of different feature maps and can reduce the number of channels of the feature maps. Table 2.7 summarizes the characteristics of the related VMMR methods introduced in this section.

2.4.5 Summary of Vehicle Make and Model Recognition Method

For the VMMR method, the goal is to find the discriminative parts of the vehicle and rely on these subtle appearance differences to recognize the model information. The general

Table 2.7: Other Representative VMMR Methods

Method	Backbone	Characteristics	Comments
Xie et al. (2015) [263]	AlexNet	Hyper class data augmentation and regularized learning	The work provides an effective data augmentation approach
Sochor et al. (2016) [236]	VGG	3D bounding box information encoding	The work provides a way of using the 3D bounding box
Zhange et al. (2016) [280]	GoogLeNet	Quadruplet loss, multitask learning	Efficient training strategy
Li et al. (2019) [163]	VGG, DenseNet	Dual cross-entropy loss	Training with the DCEL function results in fast convergence and high model performance
Chen et al. (2019) [92]	ResNet	Hierarchical multi-scale adversarial Network	A new transfer learning method for fine-grained recognition
CMP (2019) [186]	DenseNet	Channel-wise max pooling layer	CMP layer can simultaneously improve the model performance and reduce the model size

approach is to use the part detector to locate the part regions and extract their features. Some methods directly use the object detection network to detect the part regions. However, this type of method often requires extra part annotations, and the region proposal procedure is time-consuming. To speed up the region selection procedure, some methods were proposed to group the filter channels as the part detector. Inspired by the human attention mechanism, the recurrent attention-based methods iteratively extract finer or different discriminative part features. This type of method can achieve high recognition accuracy, but they often require multi-step training. The feed-forward attention-based model is relying on generating the attention map to help the model extract discriminative features along the feed-forward convolution process. Moreover, BCNN was proposed to generate pair-wise feature interactions. The feed-forward attention-based model can achieve high recognition accuracy at a relatively faster speed. We also introduced the related VMMR methods that do not belong to the three types of methods mentioned above. We hope these methods can also inspire researchers and provide potential ways to build a better recognition model.

2.5 Vehicle Re-identification

The VMMR module filters out a large number of vehicles. The rest of the images are sent to the VReID module to precisely search the target vehicle. In addition, the VReID module can also directly process the vehicles extracted by the VD module. Most of the traditional VReID methods rely on different sensors such as inductive loop detectors and wireless magnetic sensors [152]. Traditional VReID is mainly used for deriving the vehicle's travel time (time taken by a vehicle to go from one location to another) for traffic analysis and management. Considering the high extra cost and environmental sensitivity of the sensors, the sensor-based VReID method is not a good option for surveillance. A recent explosion in the use of surveillance cameras makes vision-based VReID a new demand in

the public security system. The vision-based VReID system is an essential part of ITS and has been widely used in many fields such as surveillance and social security:

1. Travel time: If the spatial-temporal information is available, then the travel time can be calculated. The travel time is useful for traffic operations, planning, and control.
2. Social security: When a vehicle is stolen or robbed by the criminal, VReID methods can help the police quickly search the target vehicle.

The vision-based VReID method finds a target vehicle from the candidates by comparing the feature distance of the anchor (query) image and candidates. For example, CNN is first used to extract the features of all images. Then, a FC layer is used to generate a n-dimensional vector for each image. Instead of using the softmax function to generate classification probabilities of each category, the n dimensional vector is used to calculate the Euclidean distance between the anchor image and other images. The image with the shortest distance from the anchor image is taken as the target image.

According to the feature type, the methods that address the VReID task can be classified into three categories: single-view appearance feature-based method, appearance and additional information-combined method, and multi-view appearance feature-based method. In the following section, we will first introduce the backbone of VReID method, and then introduce the development and comparison of each type of method.

2.5.1 Backbone of Vehicle Re-identification

VReID requires to identify and verify vehicles. Therefore, simultaneously learning the features of two or three images is a common way to train a model. The idea is to reduce the feature distance of the same vehicle and increase the feature distances of different vehicles. The Siamese Neural Network (SNN) and triplet loss function are commonly used in many models.

Siamese Neural Network

SNN [88] is widely used in VReID, due to its special structure. Bromley et al. [88] introduced SNN to solve the signature verification problem. SNN represents a special structure, and it consists of twin networks to receive two images as the inputs, which are then jointly evaluated. The loss function can simultaneously optimize the feature distances of the input image pair. The parameters of the twin networks are often shared.

Triplet loss

The triplet loss network is another backbone network. In a triplet loss network, the input is three images (triplet) $\{a, x^p, x^n\}$. In the triplet, a is the vehicle anchor (query) image, x^p (positive sample) is the same vehicle as a. In contrast, x^n (negative sample) indicates

the vehicles different than a. The feature representation of the input image x is denoted as $f(x)$. The loss function of the network is defined as:

$$L = \sum_{i=1}^{N} max\{||f(a_i) - f(x^p)||_2^2 + \alpha - ||f(a_i) - f(x^n)||_2^2, 0\}, \tag{2.19}$$

where N is the number of the samples, α is the minimum margin. The function pulls the images belonging to the same vehicle closer, and pushes the images of different vehicles away.

2.5.2 Single-view Appearance Feature-based Method

Single-view appearance feature-based methods directly use features of originally captured views (the appearance features of the input image). The vehicle similarity is determined by calculating the feature distance of the extracted single-view features of the query image and candidates. Different ways to improve this type of method are concluded as follows:

1. Feature Extraction Network Improvement: Intuitively, to get a good feature representation, many approaches have focused on improving the feature extraction ability of the model and extracting various types of appearance features (e.g., texture, color, semantic features).

2. Loss Function Improvement: Loss function is improved to learn the inter-class difference and intra-class variance.

3. Sampling Method Improvement: Models are usually trained with the triplet loss function, and the model performance is sensitive to the triplet selection. Therefore, a good sampling method can help the model converge fast and get good performance.

4. Combining other information: When the license plate information or the spatial-temporal information is available, the information can help improve the verification precision.

Feature extraction network improvement

The feature extraction network improvement is a general way to improve the model's performance. Liu et al. [175] compared the handcrafted feature-based method with the CNN-based method for VReID. Then, they proposed a Fusion of Attributes and Color feaTures (FACT) model. The model combines the handcrafted feature (color, texture) and the semantic feature (generated by CNN). In their work, the authors first calculated the rank scores of each image according to the color-based feature, texture-based feature, and semantic feature, respectively. Then the scores were summed with different weights based on the results of the experiments. The combination of handcrafted features and high-level semantic features improves the feature representation of the model.

However, simply combining different types of features has limitations in improving model performance. The combination of handcrafted features with semantic features may prevent the deep learning model from achieving higher accuracy. Jiang et al. [150] found different CNN architectures are sensitive to different attributes. They adopted the CaffeNet to extract color features, GoogLeNet to extract the vehicle model attribute, and DenseNet to learn discriminant appearance features. Instead of combining these features, they set color and model attributes as the filter conditions. The appearance feature distances are calculated only when two images have similar color and model attributes. The strategy accelerates the verification speed, and the model achieves a high reidentification accuracy. A Shortly and Densely Connected CNN was proposed in Reference [298], which was constructed by considering the advantage of VGG structure (a shortlist of convolutional layers) and DenseNet structure (the dense connection between each layer).

The models mentioned above mainly focus on extracting global features. However, vehicles of the same model have a very similar global shape, and it is difficult to distinguish different vehicles only based on global features. Some small parts, such as decorations and the window region, can provide very useful information when verifying vehicles. In Reference [133], the authors selected and labeled three vehicle parts (i.e., lights, window, brand) and trained a part detector (YOLO) to locate these parts. Vehicles were verified based on these part features and global features. Guo et al. [128] extended their coarse-to-fine model [129] by introducing a two-level attention network. First, the Spatial Transformer Network (STN) [147] was adopted to locate two vehicle parts (i.e., window screen and car-head parts). Then, they adopted a residual attention module to extract discriminative visual cues at the pixel level. This model needs to use a detection network to obtain the prior knowledge parameters for STN. Qian et al. [210] proposed a Stripe-based and Attribute-aware Network (SAN). The stripe-based branch is used to extract the part-level features through a horizontal average pooling layer. The attribute-aware branch uses two FC layers to predict the vehicle ID and vehicle model/type information. The model has a simple structure and achieves higher accuracy than other complex models.

Loss function improvement

The triplet loss function is commonly used to optimize the VReID model. However, in some situations, the triplet loss function does not perform well [170]. Figure 2.5 (a) shows two cases of distance between samples. In case 1, the triplet loss function can easily optimize the feature distance between the samples because the feature distance between the anchor and the positive sample (intra-class distance) is larger than the feature distance between the anchor and negative sample (inter-class distance). In case 2, the triplet loss function will neglect the two triplets (this situation would slow down the model convergence) because the intra-class distance is indeed smaller than the inter-class distance. The small distance between the negative sample and the positive sample may cause a failure when verifying two vehicles.

Liu et al. [170] proposed the Deep Relative Distance Learning (DRDL) model. In their

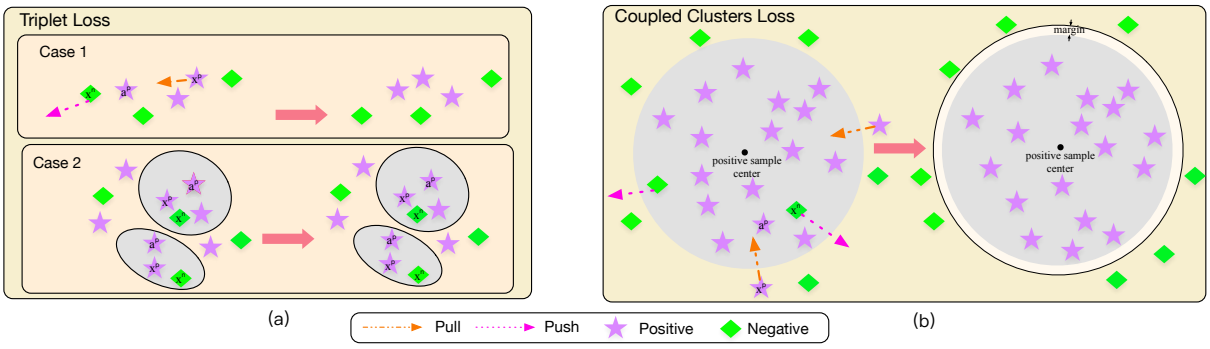


Figure 2.5: Illustration of the general triplet loss function and Coupled Clusters Loss function. (a) Triplet loss function only has the inter-class constraint, and is sensitive to the triplet selection. In case 1, the triplet loss function works fine. However, in case 2, the triplet loss function ignores optimizing the two circled triplets. (b) Coupled Clusters Loss function solves the problem of case 2, but the intra-class variance is still ignored.

work, they introduced a Coupled Clusters Loss (CCL) function:

$$L(c^p, X^p, X^n) = \sum_{i}^{N^p} max\{0, ||f(x_i^p) - c^p||_2^2 + \alpha_1 - ||f(x_*^n) - c^p||_2^2\}, \tag{2.20}$$

where N^p is the number of samples, X^p is the positive sample set, X^n is the negative sample set, c^p is the mean value of all the positive sample features (positive center point), and x_*^n is the nearest negative sample to the center point of the positive sample set. The CCL function can guarantee the negative point will be pushed away from the positive center point, as illustrated in Figure 2.5 (b). The triplet loss function has a weak constraint in distance learning. To solve the drawback of the triplet loss function, Zhang et al. [282] adopted the softmax-based classification loss function to augment the triplet loss function.

Vehicles usually have large intra-class appearance differences when the viewing angle or illumination changes. Jiang et al. [150] introduced an intra-class similarity constraint to optimize the general triplet loss function:

$$L_{in} = \sum_{i=1}^{N} max(0, ||f(a_i) - f(x^p)||_2^2 + \alpha_2). \tag{2.21}$$

Bai et al. [28] also proposed an intra-class constraint. In their work, samples of the same vehicle were further clustered into groups. An intra-class variance constraint loss function is defined as follows:

$$L_{intra}(a^{p,g}, x_g^p, x_i^p) = \sum_{g=1}^{G} \sum_{x_g^p \in S^{p,g}} max\{0, ||f(a^{p,g}) - f(x_g^p)||_2^2 + \alpha_2 - ||f(a^{p,g}) - f(x_i^p)||_2^2\}, \tag{2.22}$$

where G is the number of image groups that belong to the same vehicle. α_2 is the distance margin. $S^{p,g}$ denotes the sets of vehicle p in group g (same vehicle but in a different

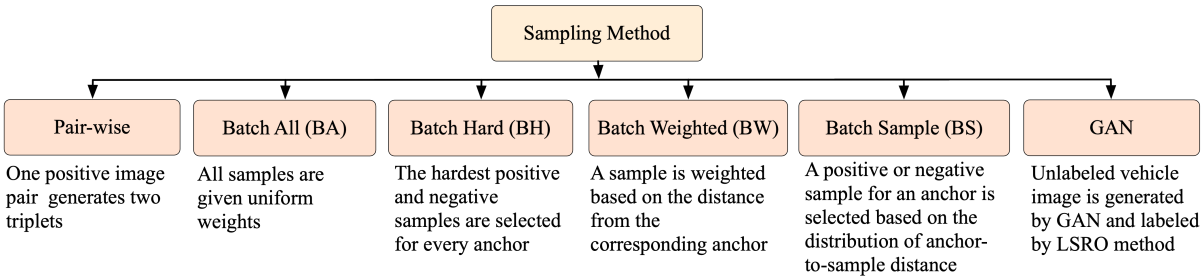


Figure 2.6: Triplet sampling method.

group). $a^{p,g} \in S^{p,g}$ is the anchor point in group g of vehicle p. x_g^p denotes the vehicle p from group g. x_i^p denotes the vehicle p from a group that different with g. Also, instead of predefining the vehicle group, an online group generation method is proposed, in which the group samples are updated during the training iteration with the k-means clustering method. The authors adopted CCL function (Equation (2.20)) proposed in Reference [170] as the inter-class similarity constraint. The proposed Group Sensitive Triplet Embedding (GS-TRE) method constraints the inter-class similarity and intra-class variance at the same time and demonstrates the importance of preserving intra-class variance. Similarly, Guo et al. [129] proposed a coarse-grained ranking loss and fine-grained ranking loss to constrain the feature distance in a coarse-to-fine manner.

Sampling method

The triplet loss function is sensitive to the triplet selection. Models can learn faster and better with a good triplet sampling method. Zhang et al. [282] proposed a Pair-wise sampling method. One positive image pair (an anchor and a positive sample) can generate two triplets, and the negative sample is from another positive image pair. This method ensures a negative sample in one triplet to be a positive sample or an anchor in another triplet. Kumar et al. [159] evaluated four sampling methods: Batch All [138], Batch Hard [138], Batch Weighted [221], and Batch Sample (BS). An illustration of these sampling methods is shown in Figure 2.6. These four sampling methods are evaluated with the model that trained with the triplet loss function and the contrastive loss function [159]. The experiment shows that the BS sampling method outperforms other methods.

A well-annotated dataset is usually hard to obtain. The model trained with the dataset collected at one place cannot be directly used in another place. To address these issues, an adoptive feature learning method was introduced in Reference [257] to sample the training data from the unlabeled video. The authors used a detector to detect the vehicles and adopted an IoU tracker [32] to track the detected vehicles. By grouping the vehicles with similar appearances (according to the Euclidean distance of the features) and vehicles that occur in the video with spatial and temporal continuity, they constructed a new dataset. Wu et al. [258] employed DCGAN [212] to generate new samples. The Label Smoothing Regularization for Outliers [288] method was adopted to model the generated vehicle category distribution. A k-reciprocal nearest neighbor method was proposed to make positive samples get a high score in the rank list.

Table 2.8: Summary of Representative Single-view Appearance Feature-based VReID Methods. F: feature extraction improvement, L: loss function improvement, S: sampling method improvement

Method	Backbone	Improve	Characteristics	Comments
FACT (2016) [175]	GoogLeNet	F	Handcrafted features (texture, color) and high-level semantic features fusion	Simply combining different types of features has limitations in model performance improvement
DRDL (2016) [170]	VGG	F, L	CCL, mixed difference network structure	Provides a unified model to achieve feature extraction and distance metric learning
Zhang et al. (2017) [282]	VGG	L, S	Optimizing triplet loss and softmax loss, triplet sampling	Simple and effective improvement; effective sampling method
GS-TRE (2018) [28]	VGG	L	Group sensitive triplet embedding, inter-class and intra-class constraint	Stable performance when vehicle pose changes; demonstrates the importance of preserving intra-class variance.
Kuma et al. (2019) [159]	-	S, L	Sampling methods evaluation, contrastive loss optimizing	BS sampling method outperforms other methods
He et al. (2019) [133]	ResNet	F	Three parts detection (YOLO), part feature and global feature fusion	Model performance is improved by accurately locating the parts of the vehicle; requires bounding box annotation
SAN (2020) [210]	ResNet	F	Horizontal average pooling layer	Simple structure; stable and high performance when vehicle pose changes

Summary

Single-view appearance feature-based methods usually have a simple model structure. With the help of a proper loss function (e.g., GS-TRE [28]), the model can achieve high reidentification accuracy, and only need the vehicle ID or vehicle attribute label to train the model. Table 2.8 summarizes the features of single-view appearance feature-based methods. To further improve the model accuracy of this type of models, some attention modules introduced in Section 2.4.2 and 2.4.3 (e.g., Reference [120, 222]) can be adopted to locate the discriminative regions of the object, and help the model distinguish subtle differences between objects.

2.5.3 Appearance and Additional Information-combined Method

The appearance-based method can hardly give optimal results due to the subtle inter-class similarity. There are various types of information (e.g., license plate information, and spatial-temporal information) that can be used to improve the model's re-identification accuracy.

Liu et al. [180] investigated whether the spatial-temporal information is useful for

verifying the similarity of two vehicles. By analyzing the space and time distance of vehicle pairs, they found the space and the time distance of the same vehicles are usually smaller than those of different vehicle pairs. According to the observation, a spatial-temporal similarity function $ST(i, j)$ of anchor image i and candidate image j is defined as:

$$ST(i,j) = \frac{|T_i - T_j|}{T_{max}} \times \frac{D(C_i, C_j)}{D_{max}}, \tag{2.23}$$

where T_i and T_j is the timestamps of i and j, T_{max} is the maximal time length among all tracks, $D(C_i, C_j)$ is the distance between camera i and camera j, D_{max} is the maximal distance among all cameras. By combining the spatial-temporal information and the license plate information, they proposed a PROgressive Vehicle Re-ID (PROVID) method [180], which progressively uses these three pieces of information. The PROVID method first uses the FACT [175] feature extraction method to extract the appearance features (i.e., texture, color, semantic) to filter out the vehicles with different visual appearances. SNN was employed to verify the license plate similarities (when the license plate is unclear) of two vehicles. Finally, the spatial-temporal information was taken into consideration to further identify the vehicles.

Shen et al. [228] proposed a two-stage framework. In the first stage, instead of simply comparing the spatial-temporal difference between two input query images, the authors generated the visual-spatial-temporal path candidate by modeling the problem as a chain Markov Random Fields model with one image as the starting state and another as the ending state. In the second stage, a SNN and path-LSTM network was proposed to verify whether two vehicles were the same vehicle. However, their model needs to pre-collect all the possible spatial paths of the same vehicles from the training set, and the model accuracy highly relies on the quality of the generated candidate visual-spatio-temporal paths. Other works [256, 294, 150] also used the spatial-temporal information.

Summary

The common weakness of appearance and additional information-combined method is that the additional information is not available in some cases, which may limit the application of these methods. Moreover, the re-identification result based on different types of information can have a conflict, so the method of weighing the different results is still an open challenge. Table 2.9 summarizes the features of the representative methods in this section.

2.5.4 Multi-view Apperance Feature-based Method

Single-view appearance feature-based methods directly compare the appearance feature distances of vehicles without considering the influence of viewpoint variation. The viewpoint variation is the major factor that drags down the model performance. The multi-view appearance feature-based method embeds the multi-view features (appearance features of a vehicle from different views) and compares the vehicle similarity in a viewpoint-invariant feature space. By learning the variation of multi-view features of the same vehicle, the model performance can be further improved.

Table 2.9: Overview of Representative Appearance and Additional Information-combined VReID Methods. SA: single-view appearance, ST: spatial-temporal information, LS: license plate information, F: feature extraction improvement

Method	Backbone	Type	Improve	Characteristics	Comments
PROVID (2016) [180, 176]	GoogLeNet, SNN	SA+ST+LS	F	Appearance feature filtering, spatial-temporal re-ranking, license plate verification	Provides a practical way of using spatial-temporal information
Shen et al. (2017) [228]	ResNet, SNN	SA+ST	F	Spatial-temporal path proposal, SNN and path-LSTM appearance verification	Requires to pre-collect the spatial path and extra time to process the path

Keypoint and viewpoint learning

Wang et al. [256] proposed an orientation-invariant feature embedding module. In this work, 20 vehicle keypoints were pre-defined. The response map of each keypoint is generated by a modified hourglass-like Fully Convolutional Network [202]. The response map generation network was trained under the supervision of ground-truth maps. Considering that some keypoints are invisible, these keypoint response maps are clustered into four sets to generate four masks of different orientations. These masks are used to create four orientation features (i.e., left, right, front, and back). Then, these orientation features are weighted by a set of scalars. These scalars are generated based on the input image. Visible orientation features are given a high weight. These weighted orientation features were combined with one global feature to generate the viewpoint-invariant feature (features are concatenated together and sent to a FC layer to generate a 256-dimensional feature vector). Vehicles were identified by calculating the Euclidean distance of the viewpoint-invariant features. Khorramshahi et al. [153] observed that using all the keypoints without selection could lead to erroneous results. Therefore, they proposed a two-stage keypoint selection framework. In the first stage, VGG was used to generate 21 coarse keypoint heatmaps. In the second stage, a two-stack hourglass network [202] was employed to refine the selected keypoints. These keypoints are grouped into eight orientation groups. The final visible keypoints selection is based on the orientation of the input image. Zhou et al. [293] proposed a CNN-LSTM to learn the non-linear transformation of viewpoints. Tang et al. [247] modified the HRNet [237] as the vehicle pose estimation backbone network to predict the keypoint coordinates (used to segment the vehicle's body) and extract heatmaps for 36 vehicle keypoints. Moreover, the authors generated a large-scale randomized synthetic dataset with a dataset synthesizer NDDS [250]. The model was trained with real and synthetic data.

The weakness of these models mentioned above is that these models usually require multi-step training. In addition, the authors cannot directly use the current dataset to train these models, extra annotations (vehicle keypoints) need to be provided. Zhu et al. [296] proposed four directional average pooling layers (i.e., pooling the features generated

by four SDCNNs along the horizontal, vertical, diagonal, and anti-diagonal directions) and a spatial normalization layer to reduce the impact of viewpoint variation. The training of the model only needs vehicle class labels and has a fast processing speed (89 FPS on a Titan X GPU).

Incorporating Generative Adversarial Network

Generative Adversarial Network (GAN) [126] contains a generator G and a discriminator D. The learning process of G and D is an adversarial process. For example, if we use GAN to generate images, then the goal of G is to generate realistic images that D cannot determine if it is fake. The goal of D is to determine whether the input image is from G or a real image. Inspired by GAN, some works [183, 182, 294] proposed adversarial learning strategies to generate training samples or train discriminators. Embedding Adversarial Learning Network (EALN) [183, 182] was proposed to generate samples related to the input image. The idea is to use the generator to generate sample $G(x)$, whose feature distance is close to the input image x, and train an embedding discriminator D_{emb} to distinguish between $G(x)$ and x (push $G(x)$ away from x). D_{emb} is considered as the feature extraction network for the embedding representation. The adversarial scheme is formulated as follows:

$$\min_{G}\max_{D_{emb}} E_{x\sim pdata(x)} max\{d(x, G(x)) - \alpha, 0\}, \tag{2.24}$$

$$d(x, G(x)) = ||D_{emb}(x) - D_{emb}(G(x))||_2^2, \tag{2.25}$$

where $pdata(x)$ represents the image distribution, $d(x, G(x))$ calculates the Euclidian distance of the embedding features between x and $G(x)$. This method was further extended to generate hard negative samples $G_{hard}(x)$, whose feature distances are closer to the input image than negative samples:

$$\beta_1 \leq d(x, G_{hard}(x)) \leq d(x, x^n) - \beta_2, \tag{2.26}$$

where x^n is the negative sample. β_1 is the minimal margin between $G_{hard}(x)$ and x, β_2 is the margin between $G_{hard}(x)$ and x^n. In addition, a cross-view adversarial learning method based on the CycleGAN [295] was proposed to generate cross-view samples. Experiments show their model can generate high quality hard negative samples and cross-view samples. Similarly, Zhou et al. [294] also proposed an adversarial multi-view feature learning method to infer the multi-view features according to the single-view features.

Summary

We summarize the features of the multi-view appearance feature-based method in Table 2.10. The multi-view appearance feature-based method has stable performance when the vehicle pose changes and can achieve higher reidentification accuracy, but these models usually have a complex model structure, and their training is a multi-stage process. We think incorporating GAN and build a more efficient structure is the future research direction. Moreover, as shown in References [247, 136], training the model with the help of synthetic data can improve model performance.

Table 2.10: Overview of Representative Multi-view Appearance Feature-based VReID Methods. ST: spatial-temporal information, F: feature extraction improvement, S: sampling method improvement, IGAN: incorporating GAN, KVL: keypoint and viewpoint learning

Method	Backbone	Improve	Characteristics	Comments
Wang et al. (2017) [256]	Hourglass	F	KVL, ST, viewpoint-invariant feature embedding, spatial-temporal regularization	An effective way to reduce the impact of viewpoint variation; requires 20 keypoints and viewpoint annotations
EALN (2019) [182]	VGG, ResNet	F, S	IGAN, embedding adversarial learning	Effectively generates high quality hard negative samples and cross-view samples; end-to-end training
Tang et al. (2019) [247]	HRNet, DenseNet	F	KVL, keypoint heatmap, segmentation, synthetic dataset training	High reidentification accuracy; training with synthetic data can also help improve the model performance
Zhu et al. (2020) [296]	SDCNN	F	KVL, quadruplet average pooling layer, spatial normalization layer	Fast speed (89 FPS); simple and effective model structure to reduce the impact of viewpoint variation; only requires vehicle's class annotation

2.5.5 Summary of Vehicle Re-identification Method

In this section, we detailed the features of each type of method. Many single-view appearance feature-based methods have shown a great performance on VeRi [175] and VehicleID [170] datasets. These methods often have a fast re-identification speed. However, these models' performance could decrease due to the illumination change and variations in vehicle pose. Some additional information, such as the spatial-temporal information and license plate information, can be incorporated to improve the model's robustness when the information is available. Considering the multi-view appearance variance and extracting local discriminative features are the major directions to improve the model performance.

2.6 Dataset

Dataset is essential to the deep learning models. However, some tasks do not have a large-scale dataset, and the models cannot obtain good generalization ability by only trained on the small-scale datasets. Therefore, most models are first pre-trained on a large-scale dataset (related to the target dataset) and then fine-tuned on the target dataset. This method is also called transfer learning [245]. There are three commonly used large-scale datasets: ImageNet [224], PASCAL VOC [117] and MS COCO [167]. Many models are first pre-trained on them and fine-tuned on the target dataset. In this section, we conclude the dataset requirements and outline the key information of the commonly used dataset for VD, VMMR, and VReID.

2.6.1 Dataset Requirements

A good vehicle-related dataset can help the model acquire a great generalization ability and systematically evaluate the model. The dataset should meet several requirements:

1. Large-scale: Deep learning is a data-driven approach. Therefore, to give the model a good generalization ability, the dataset should contain massive images for the model to learn the variation of each object.

2. Multi-view: An object in different poses can be recognized as different categories, and training the model with different views of the vehicle can increase the robustness of the model in different situations.

3. Diversified Road Environment: Vehicles in different places such as highways, countryside, or indoor parking lots may have different characteristics due to the variations in weather, illumination change, and low-heavy occlusion. A good dataset should cover images from different environments.

4. Recurrence Rate: Vehicle images with different viewpoints and backgrounds of the same vehicle should be collected in a dataset. The images of the same vehicle collected from different cameras are especially helpful for VReID models to acquire cross-camera vehicle search ability. The number of images per vehicle should be balanced.

2.6.2 Vehicle-related Dataset

Many datasets are collected for systematically and comprehensively evaluating various models for vehicle-related tasks:

- KITTI [122]: The KITTI vision benchmark suite contains a series of computer vision benchmarks such as road (lane) detection, (2d, 3d) object detection, and tracking. Images are collected by two high-resolution color and grayscale cameras, which are mounted on the vehicle. This dataset also provides depth maps captured by the Velodyne laser scanner.

- DLR-MVDA [171]: Images in this dataset are captured from an airplane at the height of 1,000 m above ground. The image resolution is 5,616 $\times$ 3,744. This dataset contains seven vehicle types.

- Stanford Cars [157]: This dataset is mainly used for VMMR. Images are collected from the website and have high resolution. There are 196 car models covering various vehicle types.

- CompCars [265]: The CompCars is a large and diversified dataset. Images are collected from public websites and surveillance cameras. The dataset provides the annotations including twelve car types, fine-grained labels (make, model and year), attributes (e.g., number of doors, maximum speed), viewpoints, car parts (e.g., headlight, fog light, dashboard, console).

Table 2.11: Summary of Vehicle-related Datasets. VD: Vehicle Detection, VMMR: Vehicle Make and Model Recognition, VMM: Vehicle Make and Model label, VTR: Vehicle Type Recognition, BBox: Bounding Box, VA: Vehicle Attribute, VReID: Vehicle Re-identification, VV: Vehicle Verification, ST: Spatial-temporal information, SEG: Segmentation. VT: Vehicle Tracking, MV: the dataset contains Multi-view images of a vehicle

Name	Application	Annotation	Basic Information
KITTI Vehicle [122]	VD, VTR	2D, 3D BBox, type	12,000 images with 40,000 labeled objects; MV
DLR-MVDA [171]	VD, VTR	BBox, type	20 images, 9,248 objects, the aerial image with a resolution of 5,616 × 3,744
Stanford Cars [157]	VD, VMMR	VMM, BBox	16,185 images of 196 classes; Train: 8,144 images; Test: 8,041 images; MV
CompCars [265]	VD, VMMR, VV, VA, VTR	VMM, BBox, VA, type	208,826 images of 1,716 vehicle models; MV
BoxCars [236]	VD, VMMR, VV	VMM, 3DBBox	63,750 images with 21,250 vehicle identities; 27 vehicle makes; MV
VeRi [175]	VReID, VTR	Vehicle ID, ST, type, color	40,395 images, 619 vehicles, and 7,021 tracks; Train: 37,781 images of 576 vehicles; Test: 11,579 images of 200 vehicles; MV
VehicleID [170]	VMMR, VReID	VMM, Vehicle ID, color	221,763 images of 26,267 vehicles; Train: 110,178 images of 13,134 vehicles; Test: 111,585 images of 13,133 vehicles; front-rear view
PKU-Vehicle [28]	VReID	Vehicle ID	10 million vehicle images with various resolutions and weather conditions; MV
Toy Car [293]	VMMR, VReID, VTR	Vehicle ID, VMM, type	30,000 toy cars images with 200 models; MV
Vehicle-1M [129]	VMMR, VReID	Vehicle ID, VMM	936,051 images of 55,527 vehicles; 400 vehicle models; Train: 844,571 images of 50,000 vehicles; Test: 91,480 images of 5,527 vehicles; front-rear view
CityFlow [248]	VT, VReID	Vehicle ID, BBox	3.25 hours of videos, 229,680 images of 666 vehicle identities; Train: 36,935 images of 333 identities; Test: 18,290 images of 333 identities, 1,052 query images; MV
VERI-Wild [183]	VReID	Vehicle ID, ST	416,314 images of 40,671 vehicle identities; Train: 277,797 images of 30,671 identities; Test: 138,517 images of 10,000 identities; MV

- BoxCars [236]: This dataset provides the 3D bounding box annotation and label annotation in a label structure of make, model, submodel and year. Images are taken from 137 surveillance cameras, and are collected for VMMR, and vehicle verification.
- VeRi [175]: This dataset is the benchmark dataset for VReID. Images are captured by 20 surveillance cameras with unconstrained angles, different illuminations, and resolutions. Vehicle images in this dataset have a high recurrence rate. The dataset provides the labels of types, colors, camera ID and vehicle ID.
- VehicleID [170]: This dataset is also the benchmark dataset for VReID. Images are captured during the daytime by multiple non-overlapping surveillance cameras. The vehicle in each image is either captured from the front or the rear. The dataset also provides the labels of colors, vehicle ID and VMM.
- PKU-Vehicle [28]: Images are captured from multiple surveillance cameras under unconstrained scenarios. The dataset is split into eight subsets to meet different requirements. This dataset is mainly used for VReID task.
- Toy Car [293]: This dataset contains vehicles with 50 different viewpoints. The camera angles are set at 30, 60 and 90 degrees to capture the vehicles. The dataset provides the vehicle type annotation. The dataset is used for VReID, VMMR and vehicle type recognition tasks.
- CityFlow [248]: This dataset provides images and original videos. Videos are taken by 40 cameras locating in diverse types of locations (e.g., highway, roadways and intersections). This dataset can be used for multi-target multi-camera vehicle tracking tasks.
- Vehicle-1M [129]: Images are captured from the front or the rear of the vehicle, across day and night. This dataset provides the vehicle model and vehicle ID annotations.
- VERI-Wild [183]: Images are captured by 174 surveillance cameras under unconstrained scenarios (i.e., different viewpoints, various backgrounds, occlusion, severe illumination and weather changes) across 30 days. This dataset is mainly used for VReID.

The general application, annotation, and features of these datasets are summarized in Table 2.11.

2.7 Open Challenges

This article outlines the recent progress and the state-of-the-art deep learning-based models that address the fundamental challenges and issues in vehicle recognition. High-precision and reliable models that have robust performance in real world environments have yet to be achieved. Other open issues to improve existing models have yet to be addressed. The

general challenges, issues, and possible directions for future research of vehicle recognition are discussed in this section.

Inter-class Similarity and Intra-class Variance. As we mentioned before, the inter-class similarity and intra-class variance are the basic challenges for different vehicle-related problems. For VD, different vehicle types can have a very similar appearance, such as truck and estate, SUV and crossover (inter-class similarity), and different vehicle models within the same vehicle type can have a large difference, such as BMW X3 and Audi Q5 (intra-class variance). For VMMR, different vehicle models need to be distinguished. Different vehicle models can have a similar appearance, such as Volkswagen Passat and Volkswagen Magotan (inter-class similarity), and different vehicles within the same vehicle model can be identified as different models when vehicles are captured under different illuminations or have different decorations (intra-class variance). For VReID, two vehicles of the same viewpoint can look more similar (inter-class similarity) than the same vehicle from different views (intra-class variance). A possible solution for these problems could be embedding the original feature into a new feature space. By learning the variation of the embedding features, the model can acquire better generalization ability.

Speed and Model Size. The recognition speed of the model is related to many factors, such as the size of the input image, the size of the model, and the hardware. However, most of the current studies have not reported the model size and the model's processing speed (computational complexity). While, these factors are the crucial factors for evaluating models. In particular, without the key information, it is difficult to determine which model is more appropriate while considering the deployment scenario. Moreover, the detection speed is especially important for VD. The frame rate of traditional surveillance cameras ranges from 10 to 30 FPS. Some high-speed cameras can even reach 60 FPS. Therefore, to achieve real-time detection, the model detection speed should be greater than 10 FPS to display detection results without delay. Current models can achieve a real-time detection speed in a powerful computer (e.g., 89 FPS on a Titan X GPU [296]); however, if we want to rely on the vehicle to achieve real-time detection, then we need to compress the model. Besides, the detection speed and precision are correlated; a model with high detection speed usually has low detection precision. Determining how to prune the model to achieve real-time detection without reducing the detection precision is still an open challenge. It has been suggested in References [131, 264] that quantizing the model is a way to shrink the network size and speed up the network.

Occlusion and Appearance Deformation Vehicles are often partially or severely occluded by front vehicles, or pedestrians. This occlusion could seriously affect detection and recognition precision. Only a few works have tried to solve the occlusion problem. The vehicle has a symmetrical shape. A possible solution is to take advantage of this special characteristic. For example, when the vehicle is partially occluded, one can use a keypoint location method to predict the location of the whole vehicle [160, 132, 111]. In addition, using a proper data augmentation method (e.g., random erasing [291, 169]) during training can also make the model robust to occlusion. Moreover, the appearance of the vehicle would change due to a traffic accident. For a deep learning-based model, the influence of different deformation degrees on the model performance is still unclear, which could be a potential research topic.

Evaluation Under Different Environmental Conditions. As we mentioned before, diversified illuminations, weather conditions, and road conditions could affect the performance of vehicle recognition models. However, to the best of our knowledge, there is no paper that comprehensively evaluates the performance of different deep learning-based vehicle recognition models under different environmental conditions. In addition, different models need to be comprehensively and fairly evaluated under the same runtime environment (hardware, deep learning platform). Therefore, this could also be a potential research topic.

Multi-modal Sensing. In this article, we only introduced vision-based vehicle recognition works. The detection and identification performance of the VAVR system can be affected by severe weather conditions, such as heavy snow and fog. Overcoming the limitations of vision sensors by combining other sensors such as lidar and radar can benefit the VAVR system. Also, the prediction results obtained based on different information may conflict. How to weigh the importance of the different results in different environments is worth studying. He et al. [137] has made attempts to achieve the fusion of vision and radar information.

Online Training. One of the main disadvantages of the deep learning-based method is that every time a new vehicle model is produced, we need to retrain the whole model. The repeated retraining will waste time and resources. It is imperative to figure out how to update the model without retraining the whole model. We believe the RL method could be a possible solution.

Dataset. A comprehensive dataset that meets the requirements we mentioned in Section 2.6.1 is essential to the deep learning-based model. Most of the images in the current VD dataset were collected under general environmental conditions. Therefore, there is an urgent need for the VD dataset that collects images under different environmental conditions (especially harsh weather conditions). Also, there is a lack of a multi-VD dataset labeled with general vehicle types and fine-grained vehicle model information. With these datasets, researchers can explore more potential research topics to enrich the ITS and Smart City applications. Meanwhile, the model's robustness can be verified with the dataset collected from different scenarios.

VMMR. Current VMMR methods focus on single-vehicle recognition. A possible research topic could be multi-target fine-grained VD and recognition. Also, to improve the model practicality, a multi-stage approach can be adopted, for example, in the first stage, the VMMR method can first identify the make and model of the vehicle, then recognize its year of manufacture.

VReID. Vehicle appearance is more likely to be changed after a long time interval and a large space separation. When identifying and verifying two vehicles, a change in appearance can easily cause a malfunction. This highlights the importance and difficulty of building a long-term reliable VReID model. In addition, for a dataset with N images, there could be $O(n^3)$ triplets. Training a model with all triplets is time-consuming and inefficient since only a minority of triplets are useful. Therefore, designing a suitable sampling method or a proper loss function could be the future direction of research.

2.8 Summary

In this chapter, we provide a systematic review of the literature using deep learning-based methods to solve vision-based VD, VMMR, and VReID problems. For VD, on-road and aerial image-based VD methods are achieved by improving the current two-stage and one-stage general object detection methods. For VMMR, there are three major approaches: part detector-based, feed-forward attention-based, and recurrent attention-based methods. For VReID, earlier works found the target vehicle by directly comparing the single-view appearance features. Recent works embedded appearance features to a viewpoint-invariant feature space to perform the comparison. Other methods combined the spatial-temporal information and license plate information to improve the model performance. We also introduced the dataset requirements and characteristics of each dataset. The open challenges and potential directions for further research are discussed comprehensively.

Chapter 3

Recurrent Attention Unit

In this chapter, we propose the Recurrent Attention Unit (RAU), which is designed as a modular attention unit that can be applied to different convolutional layers without changing the original structure of the standard CNN. This design allows the transfer learning approach to be instantly applied to the proposed models. When solving the VMMR task, RAU seamlessly enhances the performance of standard CNN architectures (such as ResNet). The content of this chapter has been published in *An AI-based Visual Attention Model for Vehicle Make and Model Recognition*.

3.1 The Proposed Approach and Assumption

RAU can be applied on different layers of standard CNN architectures to extract prominent information from different scales. Each RAU takes the feature maps generated by the convolutional layer and the attention state generated by the previous RAU as inputs. Afterward, the unit produces a new state for the next RAU. The mechanism is not only extracting discriminative information from different layers but also combining them. This process allows the model to recognize an object by evaluating the discriminative features of different resolutions.

3.1.1 Recurrent Attention Unit

The internal structure of RAU can be concluded as three parts: feature integration, attention mask generation, and attention state generation. For the feature integration submodule, we integrate the features from the feature maps and the attention state. First, the feature maps are processed by a convolutional layer to generate $M_h^l \in \mathbb{R}^{kn \times h^l \times w^l}$:

$$M_h^l = W_{uh}^l * H^l, \tag{3.1}$$

where $H^l \in \mathbb{R}^{c^l \times h^l \times w^l}$ are the feature maps for the lth RAU, $l \in [1, ..., L]$. The height and width of the feature maps are denoted by h^l and w^l, c^l is the channel of the feature

maps. The number of attention masks is denoted by k, n is the number of vehicle model categories. The whole convolution process of the convolutional layer is denoted by '$*$'. W_{uh}^{l} represents the overall parameters of the convolutional layer.

To make the attention state $S^{l-1} \in \mathbb{R}^{kn \times h^{l-1} \times w^{l-1}}$ be compatible with the parameters of current RAU, a convolutional layer is utilized to reduce the spatial size of the attention state S^{l-1} and we use another convolutional layer to generate M_s^l:

$$S'^{l-1} = W_s^l * S^{l-1}, \tag{3.2}$$

$$M_s^l = W_{us}^l * S'^{l-1}, \tag{3.3}$$

where $S'^{l-1} \in \mathbb{R}^{n \times h^l \times w^l}$, $M_s^l \in \mathbb{R}^{kn \times h^l \times w^l}$. Then M_h^l and M_s^l are combined together to generate the feature representation U^l:

$$U^l = tanh(M_h^l + M_s^l + b_u), \tag{3.4}$$

where $U^l \in \mathbb{R}^{kn \times h^l \times w^l}$, $tanh$ introduces nonlinearity to the sum of M_h^l and M_s^l, b_u is the bias.

For the attention mask generation submodule, whether the attention mask is generated by considering features from the feature maps and the attention state is what we want to ensure. Therefore, H^l and S'^{l-1} are sent to two convolutional layers to generate k learnable single-channel attention masks respectively, and then they are added together:

$$C^l = \sigma(W_{ch}^l * H^l + W_{cs}^l * S'^{l-1} + b_c), \tag{3.5}$$

where $C^l \in \mathbb{R}^{k \times h^l \times w^l}$ will select the information of the attention state S'^{l-1}. These attention masks focus on the discriminative regions of S'^{l-1}. The sigmoid function is denoted by σ.

The new attention state $S^l \in \mathbb{R}^{kn \times h^l \times w^l}$ is the sum of the selected features and U^l:

$$S^l = C^l \odot S'^{l-1} + U^l, \tag{3.6}$$

where $\odot$ denotes the element-wise product. The feature selection operation is done by repeating each mask for each of the feature channels of S'^{l-1} and conducting element-wise production. The attention state S^l represents the features of n vehicle models on k discriminative locations. In VMMR, the module should extract distinctive information of the object. We do not want to discard (weaken) the semantic information of the same region in the feature representation U^l. Therefore, U^l is directly added in Eq. (3.6), rather than adding the $(1 - C^l) \odot U^l$ as in [181].

The difference between different vehicle models is quite subtle. Feature maps of different scales can provide different levels of information. Therefore, it is useful to collect and combine sufficient discriminative features from multi-scale regions. Therefore, we deploy the RAUs to different convolutional layers to receive the feature maps of different scales. The attention state S^l generated by the last RAU represents the integrated part features. The Global Average Pooling layer [134] is used to pool these part features, resulting a $kn \times 1 \times 1$ vector. The vector represents the scores of n vehicle models selected by k

Table 3.1: The detailed information of the weights of the convolutional layers in RAU

weight name	size of the kernel	stride	number
W_{uh}	1×1	1	kn
W_{ch}	3×3	1	k
W_s	3×3	2	n
W_{us}	1×1	1	kn
W_{cs}	3×3	1	k

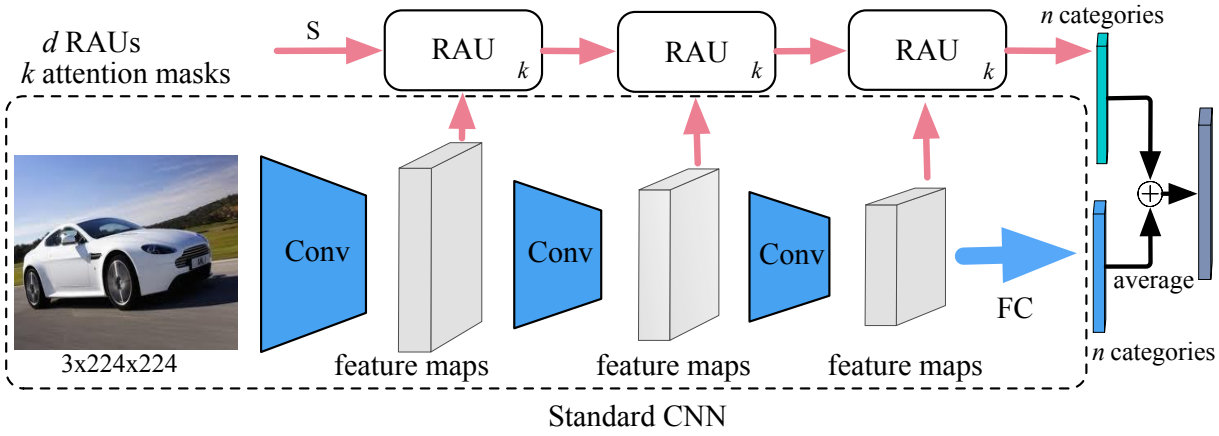


Figure 3.1: The framework of the proposed model. Three Recurrent Attention Units (RAUs) are applied to the original CNN architecture at three different layer groups. Each RAU contains k attention masks. The final results come from the original CNN and the final RAU.

attention masks. Then, the vector is processed by the FC layer to generate the classification scores.

The final recognition result is the average of the results of the standard CNN and the last RAU. The loss function is defined as:

$$L = \frac{1}{2}\sum_{i=1}^{n} \hat{Y}_i log Y_i + \frac{1}{2}\sum_{i=1}^{n} \hat{Y}_i log P_i, \tag{3.7}$$

where $P \in \mathbb{R}^n$ is the predicted classification probabilities from the last RAU, Y_i is the predicted probability for class i from the standard CNN, $\hat{Y}$ is the one-hot encoded ground truth label vector. This loss function simultaneously optimizes the global and part feature extraction abilities of the model. The detailed information about the weights of the convolutional layers in RAU are provided in Table 3.1.

3.1.2 Using RAU with ResNet

RAU is compatible with most standard CNN architectures. We choose two strong baselines ResNet50 and ResNet101 [134] as our base models. By deploying RAUs to these two base models, we get two new models, ResNet50-RAU and ResNet101-RAU. A three-RAU

Table 3.2: The statistics of three VMMR datasets. BBox stands for bounding box

Datasets	Category	Train	Test	BBox
Stanford Cars [157]	196	8,144	8,041	√
CompCars [265]	431	16,016	14,939	√
CompCars Surveillance [265]	281	31,148	13,333	

(a) Stanford Cars (b) CompCars SV (c) CompCars

Figure 3.2: Samples from (a) Stanford Cars dataset, (b) CompCars Surveillance dataset and (c) CompCars dataset in our experiments.

structure ResNet-RAU is presented in Figure 3.1. ResNet contains several groups of convolutional layers, which are called layer groups. Each layer group contains a set of Residual Network [134], which contains convolutional layers with the same number of channels. ResNet50 and ResNet101 both consist of four convolutional layer groups, but each layer group contains a different number of convolutional layers. The sizes of the feature maps generated by theses four layer groups are $256 \times 56 \times 56$, $512 \times 28 \times 28$, $1024 \times 14 \times 14$, and $2048 \times 7 \times 7$. We deploy RAUs to the end of the last three layer groups to test the effectiveness of our proposed module. The recognition ability of each RAU is controlled by the number of attention masks k. It is worth mentioning that each attention mask can focus on different locations. The final classification result of the whole model is the average of the results from the global stream (standard CNN) and the part stream (RAUs). The related experiments are shown in Section 3.2.2.

3.2 Experiments

We conduct experiments on three benchmark vehicle datasets, including Stanford Cars [157], CompCars [265] and CompCars Surveillance [265]. The detailed information of these datasets is provided in Table 3.2. Samples from these datasets are shown in Figure 3.2.

3.2.1 Implementation Details

Our method is implemented with PyTorch [206]. The two standard CNN architectures are first pre-trained on ImageNet dataset [235], later fine-tuned on the target datasets. For the original ResNet architecture, only the last FC layer is modified to adapt to each dataset. For all the experiments, the models are trained using Stochastic Gradient Descent (SGD) with a weight decay of 0.0005 and a momentum of 0.9. Models are trained for 120 epochs. The initial learning rate is set to 0.01. After 40 epochs, the learning rate is reduced by a factor of 0.5 every 20 epochs. The image is resized to 224×224. Data augmentations such as random crop and horizontal flip are applied. For the last 40 epochs, images will be cropped with the help of the bounding box when training the model on the CompCars dataset. The weights of the convolutional layers in RAU are initialized by using the method presented in [134], biases are initialized to zero.

3.2.2 Ablation Study

We evaluate the influence of two adjustable parameters of our model on the Stanford Cars dataset following a similar procedure presented in [222]. We conduct experiments to show the effect of the number of RAUs d and the number of attention masks k. To capture more target information contained in features of different scales, we need to deploy at least two RAUs to combine these features. The maximum number of RAUs that can be added is determined by the structure of the base model. For example, ResNet contains four layer groups, and three of them generate mid-level and high-level features, which are the main useful features for recognizing an object, since the information carried by the lower layers is very fundamental, and features of higher layers are more class-specific [255]. Therefore, the maximum value of d is 3 in our experiment. The value of k is set from 1 to 4.

The number of parameters and floating point operations (FLOPs) [197] are usually used to measure the computational complexity of deep learning models. The experimental results, including the accuracy and computational complexity of ResNet50-RAU with different d and k, are presented in Table 3.3. We can observe that our model can achieve a better result than the previous state-of-the-art methods (shown in Table 3.4) when combining features extracted from feature maps of different scales. Moreover, when we increase k to 2 ($d = 2$), our model achieves the best recognition result 93.57%. In Table 3.3, we can also observe that the computational complexity of the models has increased, but our models still have the ability to process images with real-time processing speed. For example, when running on a workstation with a GTX1080Ti GPU, the processing speed of ResNet50 ($d = 2, k = 2$) is 128 frames per second (FPS). For the next part of our experiments, we set d to 2 and k to 2 for ResNet50-RAU as these values give the best recognition result. Since ResNet50-RAU and ResNet101-RAU have a similar structure, we use the same parameter setting for ResNet101-RAU.

Table 3.3: The influence of the number of RAUs d, and the number of attention masks k for each RAU on the Stanford Cars dataset

	ResNet50-RAU					
	$d = 2$			$d = 3$		
	A	P	F	A	P	F
$k = 1$	93.42	25.0	4.20	93.30	25.5	4.36
$k = 2$	**93.57**	26.0	4.28	93.40	27.0	4.59
$k = 3$	93.32	27.1	4.36	93.41	28.5	4.83
$k = 4$	93.26	28.1	4.44	93.35	30.1	5.06
baseline	ResNet50	91.78		P = 23.9	F = 4.12	

Abbreviations: A: Accuracy (%); P: The number of parameters ($\times 10^6$); F: The number of FLOPs ($\times 10^9$).

3.2.3 Comparison with State-of-the-Art Methods

Performance on Stanford Cars

We compare the proposed models with recent state-of-the-art methods. The results are listed in Table 3.4. In this table, the results of the *previous* part are cited from the original papers, and the results of *baseline* models and our models are derived from the experiments that are conducted based on the experimental setting we mentioned before. Without adopting the bounding box annotation at test time, our model ResNet101-RAU achieves 91.47% recognition accuracy, which outperforms the baseline ResNet101 with a clear margin (4.14% relative gains), and it is also a comparable result with M-CAN (91.5%), which adopted the Class Activation Map (CAM) to select object features [292]. RA-CNN [120] and MA-CNN [287] crop out the part regions of the object according to their attention modules, and these models are trained with images of size 448 × 448. These part regions and large scale images provide more detailed information, which allows these models to produce higher results of 92.5% and 92.8%, respectively. PA-CNN [156] adopts the co-segmentation and alignment methods and tests the model by adopting the bounding box annotation, which achieves 92.8% accuracy. Under a similar experiment setting, the baseline ResNet50 and ResNet101 achieve 91.78% and 92.45%, respectively. By adopting our attention selection method, the recognition accuracies are improved from 91.78% to 93.57% for ResNet50-RAU and from 92.45% to 93.81% for ResNet101-RAU. The evaluation shows that the proposed RAU advances the performance of ResNet for the VMMR task. The previously reported best recognition accuracy 93.1% is achieved by ResNet101-SWP [141] and FCAN [179], ResNet101-RAU achieves a better recognition result with 0.71% relative improvement.

Performance on CompCars

The recognition results on the CompCars dataset are summarized in Table 3.5. The two baselines ResNet50 and ResNet101 obtain 96.73% and 97.41% accuracy, respectively. By adopting our RAU, the values of the accuracy of the proposed ResNet50-RAU and ResNet101-RAU are boosted to 97.60% and 97.84%, respectively. The improvement shows

Table 3.4: Comparison of the recognition results on the Stanford Cars dataset, $\surd$ means testing with bounding box annotation, N/A indicates that bounding box annotation is not used

Type	Model	Test Anno.	Accuracy (%)
Previous	DVAN [284]	N/A	87.1
	WARN [222]	N/A	90.0
	BCNN (448) [168]	N/A	91.3
	M-CAN [290]	N/A	91.5
	RA-CNN (448) [120]	N/A	92.5
	MA-CNN (448) [287]	N/A	92.8
	ResNet50-SWP [141]	$\surd$	92.3
	PA-CNN [156]	$\surd$	92.8
	FCAN [179]	$\surd$	**93.1**
	ResNet101-SWP [141]	$\surd$	**93.1**
Baseline	ResNet50	N/A	86.78
	ResNet101	N/A	87.33
	ResNet50	$\surd$	91.78
	ResNet101	$\surd$	92.45
Ours	ResNet50-RAU	N/A	90.30
	ResNet101-RAU	N/A	91.47
	ResNet50-RAU	$\surd$	93.57
	ResNet101-RAU	$\surd$	**93.81**

Table 3.5: Comparison of the recognition results on the CompCars dataset

Type	Model	Accuracy (%)
Previous	AlexNet [265]	81.9
	OverFeat [265]	87.9
	GoogLeNet [265]	91.2
	ResNet50-SWP [141]	97.5
	ResNet101-SWP [141]	**97.6**
Baseline	ResNet50	96.73
	ResNet101	97.41
Ours	ResNet50-RAU	97.60
	ResNet101-RAU	**97.84**

our models have a stable performance on recognizing vehicle models. The recently proposed ResNet101-SWP [141] achieves 97.6% accuracy on this dataset, ResNet101-RAU surpasses it by a small margin of 0.24%, achieving the best recognition result on the CompCars dataset.

Performance on CompCars Surveillance

For this dataset, we only trained our models for 50 epochs. The initial learning rate is set to 0.001. The learning rate is divided by 0.5 at 30th and 40th epoch. We use this dataset to test the performance of our models in a practical environment. The recognition results on the CompCars Surveillance dataset are summarized in Table 3.6. A coarse-to-fine (CF) [118] method was proposed to iteratively extract global and local vehicle features, achieving 98.63% accuracy. Lightweight Convolutional Neural Network (LWCNN) [279] is

Table 3.6: Comparison of the recognition results on the CompCars Surveillance dataset

Type	Model	Accuracy (%)
Previous	AlexNet [265]	98.0
	OverFeat [265]	98.3
	GoogLeNet [265]	98.4
	CF [118]	98.63
	LWCNN [279]	98.71
	FF-CMNET [271]	**98.89**
Ours	ResNet50-RAU	98.83
	ResNet101-RAU	**98.90**

a simplified VGG model and is trained with a combined training strategy. The lightweight VGG achieves 98.71% accuracy. FF-CMNET [271] uses two separate networks to extract the upper part and down part features of a vehicle, and it achieves 98.89 % accuracy. Our proposed models ResNet50-RAU and ResNet101-RAU achieve results of 98.83% and 98.90%, respectively, which demonstrate our models can still obtain excellent results on the surveillance dataset.

3.3 Summary

In this paper, we propose the Recurrent Attention Unit (RAU) to enhance the performance of standard CNN architectures for the VMMR task. RAU can help the model extract the most discriminative features of an object. RAU is easy to be added to most standard CNN architectures and can be trained end-to-end. Extensive experimental results demonstrate the effectiveness of fusing prominent features of different scales. With the help of the Vehicle-to-Everything (V2X) technology, the accurately identified vehicle information can be transmitted to the agents, which will help the police find the criminal and increase the region's security. In the future, we will try to simplify the structure of RAU and reduce the size of the model, so that the proposed models can have excellent performance under different hardware conditions. We will also test the performance of RAU when combining it with other models.

Chapter 4

Lightweight Recurrent Attention Unit

In this chapter, we simplify the structure of RAU and propose the Lightweight RAU (LRAU). We add LRAUs to ResNet and VGG to construct two VMMR models, ResNet50-LRAU and VGG16-LRAU. In addition, we add LRAUs to the detection network SSD to construct a one-stage Vehicle Detection and Fine-grained Recognition (VDFR) model, SSD-LRAU. The proposed SSD-RAU can generate the bounding box and fine-grained vehicle information simultaneously. The content of this chapter has been published in *A Lightweight Visual Attention Model for Fine-Grained Vehicle Recognition.*

4.1 The Proposed Approach

Convolutional filters in different layers can have relatively different receptive field sizes, and these filters focus on extracting different types of features. In the lower layer of a CNN, convolutional filters have a relatively small receptive field (224×224 vs. 3×3), and these convolutional layers mainly focus on extracting the fundamental features such as corners and lines. In higher layers, convolutional filters have a larger receptive field (14×14 vs. 3×3), and they focus on more class-specific features (e.g., the general shape of the object). By visualizing the convolutional filters [276], we can observe such a working pattern. However, during the feed-forward extracting process, we think some discriminative features may get lost. Therefore, to retain the discriminative features, we could establish a connection to combine fine-grained features with coarse features. This is achieved by generating attention masks based on the current input features maps and preceding attention features, and using these attention masks to select discriminative features from feature maps. Finally, the selected features are fused with the preceding attention features. These fused discriminative features help recognize an object.

The designed LRAU is a modular attention unit and it can be easily added to different standard CNN architectures such as ResNet [134], VGG [233] and InceptionV3 [243] to enhance their performance on solving the VMMR task. In the following, we will introduce the structure of LRAU comprehensively.

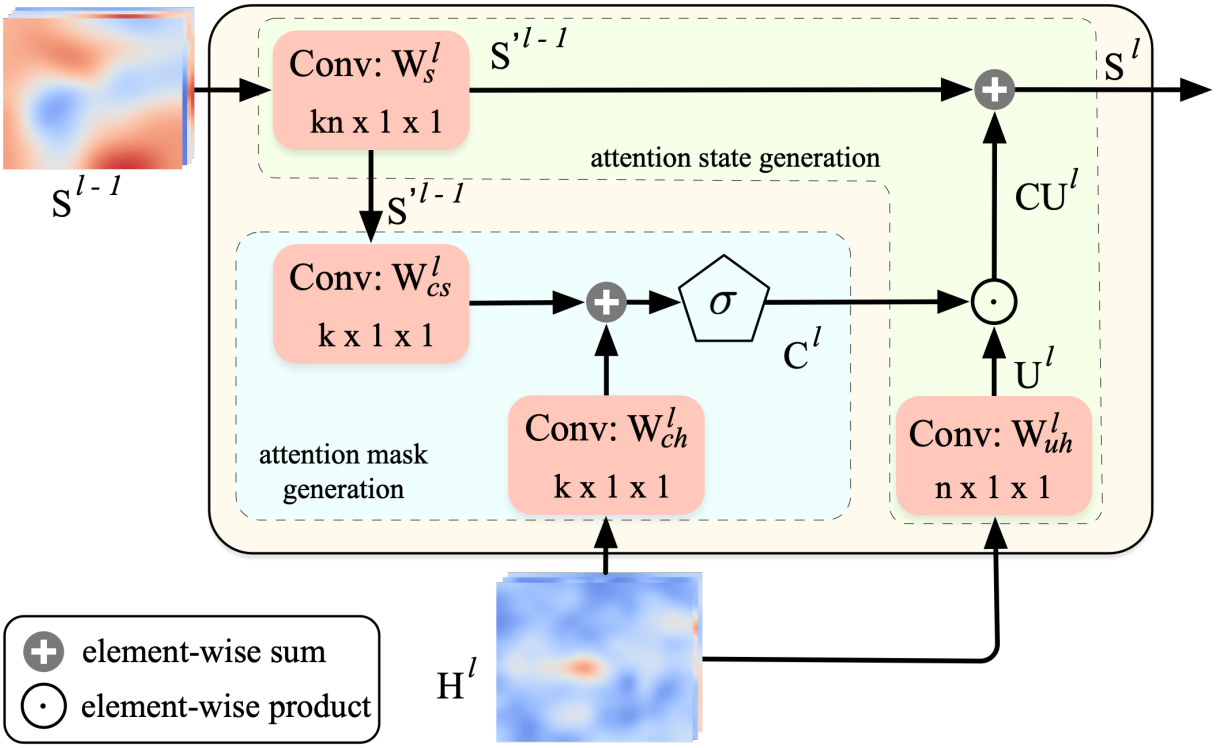


Figure 4.1: The detailed internal structure of the Lightweight Recurrent Attention Unit (LRAU). The convolutional layer is denoted by *Conv*. For simplicity, biases are omitted.

4.1.1 Lightweight Recurrent Attention Unit

The detailed structure of our proposed LRAU is illustrated in Figure 4.1. The standard CNN backbone takes the image as input to generate multi-scale feature maps along the feed-forward process. The inputs for each LRAU are the feature maps H^l and the preceding attention state S^{l-1}. The initial attention state is set to 0. After a series of operations on the feature maps and attention state, LRAU will output a new attention state S^l. Each attention state retains the discriminative features selected by the attention masks.

The internal structure of LRAU can be divided into two parts: attention mask generation, and attention state generation. For the attention mask generation submodule, the attention masks are generated based on the current input feature maps and the preceding attention state. First, the input feature maps and the resized attention state are processed by two convolutional layers to generate k attention masks, respectively. Each mask highlights different discriminative regions of the image:

$$C^h = W^l_{ch} * H^l + b_{hc}, \tag{4.1}$$

$$C^s = W^l_{cs} * S'^{l-1} + b_{sc}, \tag{4.2}$$

where $H^l \in \mathbb{R}^{c^l \times h^l \times w^l}$ represents the feature maps for the lth LRAU, $l \in [1, ..., L]$. The height and width of the feature maps are denoted by h^l and w^l, c^l is the channel of the feature maps, S'^{l-1} is the resized attention state. The attention state generation will be

introduced in the following. W_{ch}^{l} and W_{cs}^{l} are the overall parameters of the convolutional layers. The whole convolution process of the convolutional layer is denoted by '$*$'. By directly using the 1×1 convolutional layer to process the feature maps, the value of a location in the attention map is determined by evaluating the importance of each feature map channel in that location. During training, the filter will learn to assign a high weight to the important channels and a lower weight to the less important channels. Then, the attention masks C^h and C^s are combined through the element-wise summation. The values of the masks are normalized between 0 to 1 by the sigmoid function:

$$C^l = \sigma(C^h + C^s), \tag{4.3}$$

where $C^l \in \mathbb{R}^{k \times h^l \times w^l}$, σ is the sigmoid function. C^h and C^s highlight different discriminative regions since they are generated based on different features. By summing these two types of attention masks, the combined attention masks can focus on a wider discriminative region, while highlighting the keypoint regions.

For the attention generation submodule, we first use a convolutional layer to convolve the feature maps to have n channels:

$$U^l = W_{uh}^{l} * H^l + b_u, \tag{4.4}$$

where $U^l \in \mathbb{R}^{n \times h^l \times w^l}$, n is the number of vehicle categories. In the convolutional layer, each filter needs to have the same channels as the input feature maps; by using n filters, we select the features for n vehicle models. Therefore, each channel of U^l can be seen as the compressed feature for one vehicle model. Then, the attention masks are used to select the discriminative features. The feature selection is done by repeating each mask for each feature channel of U^l, and then performing the element-wise production:

$$CU^l = C^l \odot U^l, \tag{4.5}$$

where $\odot$ denotes the element-wise product. The selected features $CU^l \in \mathbb{R}^{kn \times h^l \times w^l}$ can be interpreted as the features of n vehicle models selected by k attention masks.

The difference among different vehicle models is quite subtle, so it is useful to collect sufficient prominent features from multi-scale regions to make correct identification decisions. Let $S^{l-1} \in \mathbb{R}^{kn \times h^{l-1} \times w^{l-1}}$ denote the attention state generated by the preceding LRAU. To make the attention state S^{l-1} compatible with the currently selected features CU^l, S^{l-1} should have the same size as the input feature maps. Therefore, a convolutional layer is utilized to reduce the size of the attention state S^{l-1}:

$$S'^{l-1} = W_s^l * S^{l-1} + b_s, \tag{4.6}$$

where $S'^{l-1} \in \mathbb{R}^{kn \times h^l \times w^l}$.

The new attention state is generated by combining the preceding attention state with the currently selected features:

$$S^l = CU^l + S'^{l-1}. \tag{4.7}$$

The Attention Residual Learning (ARL) [252] introduced the idea of using attention masks to select features, and then combine the selected features with the original feature

Table 4.1: The detailed information of the convolutional filters in LRAU for VMMR. n is the number of vehicle categories, and k is the number of attention masks

weight name	size of the kernel	stride	number
W_{uh}	1×1	1	n
W_{ch}	1×1	1	k
W_{cs}	1×1	1	k
W_s	1×1	2	kn

maps. Inspired by ARL, we also adopt the idea of using the attention mask, but our attention mask generation is different from ARL. Instead of summing the original feature maps, we fuse the selected features with the attention state generated by the preceding LRAU.

The attention state S generated by the last LRAU represents the integrated discriminative part features. A Global Max Pooling (GMP) layer is used pool S to select the prominent values over the spatial location, obtaining a $kn \times 1 \times 1$ vector. During training, GMP causes the gradients to be backpropagated to the prominent location. Finally, a FC layer is used to generate the classification scores based on these discriminative part features.

The final recognition results combine both the results from the standard CNN architecture and the last LRAU. The loss function is defined as:

$$L = \frac{1}{2}\sum_{i=1}^{n} \hat{Y}_i log Y_i + \frac{1}{2}\sum_{i=1}^{n} \hat{Y}_i log P_i, \tag{4.8}$$

where $P \in \mathbb{R}^n$ is the predicted classification probabilities from the last LRAU, Y_i is the predicted probability for class i from the standard CNN architecture, $\hat{Y}$ is the one-hot encoded ground truth label vector. This loss function simultaneously optimizes the global and part feature extraction abilities of the model. Therefore, our proposed model can be trained end-to-end. The detailed information of the weights of the convolutional layers in LRAU is provided in Table 4.1.

4.1.2 Using LRAU with Standard CNNs

LRAU is compatible with most standard CNN architectures. To fairly compare our models with other state-of-the-art models and prove the compatibility of LRAU, we choose three popular CNN architectures, namely ResNet50, VGG16 and InceptionV3, as our baselines. By deploying LRAUs to these three baselines, two VMMR models are generated. We denote the two models as ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU. The implementation details and the variants to the three baselines are introduced below.

The original ResNet50 consists of the feature extraction network and classifier. ResNet only uses one FC layer as the classifier, and we only modify the FC layer to predict classification scores that match the number of classes in the target datasets. The modified

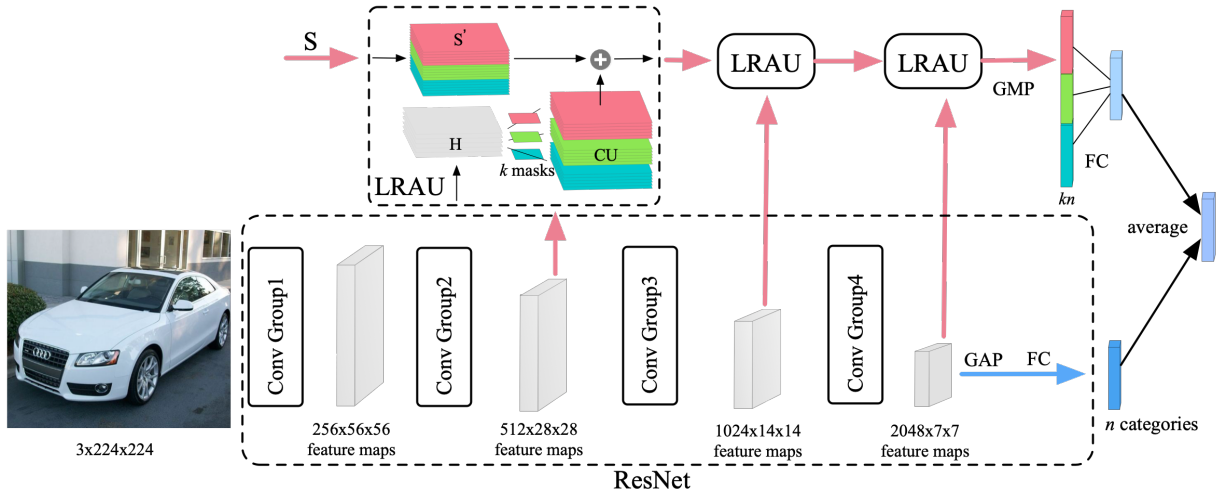


Figure 4.2: The framework of the proposed VMMR model. In this figure, we use ResNet as the backbone, three LRAUs are added to the ResNet at three different convolutional layer groups. Each LRAU contains three attention masks. In the first LRAU, we show the process of attention state generation. For a clear visualization, the attention mask generation process is omitted.

baseline is still referred to as ResNet50. The feature extraction network of ResNet50 consists of four convolutional layer groups. Each convolutional layer group contains a different number of convolutional layers (batch normalization layers, activation layers). The sizes of the feature maps generated by the four group layers are $256 \times 56 \times 56$ ($group1$), $512 \times 28 \times 28$ ($group2$), $1024 \times 14 \times 14$ ($group3$), and $2048 \times 7 \times 7$ ($group4$). ResNet50-LRAU is constructed by deploying LRAUs after each convolutional layer group of ResNet50, and LRAUs receive the feature maps generated by these convolutional layer groups as one of the inputs. The branch that contains LRAUs is called the part feature extraction branch. The recognition ability of the part feature extraction branch is controlled by the number of LRAUs d and the number of attention masks k for each LRAU. We prioritize adding LRAUs to the higher layers. For example, when d is set to 2, LRAUs are added after $group4$ and $group3$. Figure 4.2 shows a three-LRAU structure of ResNet50-LRAU, LRAUs are added after $group4$, $group3$ and $group2$.

The original VGG uses three FC layers as the classifier. We remove these three FC layers, since they take most of the parameters of VGG. Then, we add the global average pooling layer at the end of VGG to pool the $512 \times 7 \times 7$ feature maps to a $512 \times 1 \times 1$ vector. Finally, we add one FC layer to classify the vector. The modified VGG is referred to as VGG16-GAP. VGG uses the max pooling layer to reduce the size of the feature maps, so LRAU can be deployed around each max pooling layer to receive feature maps of different scales. More specifically, we deploy LRAUs after $Conv3\_3$, $Conv4\_3$, $Conv5\_3$ (i.e., before the reduction of the size of the feature maps), and after the last max pooling layer of VGG16-GAP to construct VGG16-LRAU.

InceptionV3 is an improvement of GoogLeNet[244]. For the original InceptionV3, we remove the auxiliary classifiers and modify the FC layer to fit the target datasets. Incep-

Table 4.2: The detailed information of the convolutional filters in LRAU for VDFR, c is the channel of the feature maps, and k is the number of attention masks

weight name	size of the kernel	stride	number
W_{uh}	1×1	1	c
W_{ch}	1×1	1	k
W_{cs}	1×1	1	k
W_s	1×1	2	kc

tionV3 mainly consists of different types of Inception modules. Each Inception module has different branches and each branch is the combination of convolutional layers with different kernel sizes (e.g, 1×1, 3×3, 1×3, 1×7). For the Inception module, feature maps generated by its branches are finally concatenated together and then sent to the following Inception module. Our LRAUs are added to receive the concatenated feature maps. The original input size of InceptionV3 is 299×229, whereas the input size is 224×224 in our experiment. Therefore, we change the size of the last pooling layer accordingly. The sizes of feature maps we used are $192 \times 52 \times 52$, $288 \times 25 \times 25$, $768 \times 12 \times 12$, and $2048 \times 5 \times 5$. Compared with ResNet50 and VGG16, we can observe that the size of these feature maps is reduced by a different ratio. Thus, we change the kernel size of W_s to 3×3 to increase compatibility among LRAUs. We name this model InceptionV3-LRAU. Such design guarantees that LRAU receives multi-scale feature maps, and we can conduct a fair comparison of the proposed ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU under the same setting.

4.1.3 Vehicle Detection and Fine-grained Recognition

Generally, when an image contains multiple vehicles, an object detection method is first used to locate these vehicles. Then, vehicles are cropped out according to the predicted bounding box, and these vehicles are sent to the recognition model to perform vehicle make and model recognition. We refer to this type of method as the two-stage VMMR method. Alternatively, we can also build a model to locate the vehicles and recognize the vehicle information simultaneously (one-stage). We add LRAUs to an object detection model SSD [173] to construct a one-stage VDFR model.

In SSD, feature maps of different scales are sent to the convolutional layers to generate the default bounding box adjustments and the classification scores of general objects (e.g., car, bicycle, cat). To improve the fine-grained recognition performance of SSD, feature maps are sent to LRAUs to generate the attention states. Then, these attention states are sent to the corresponding convolutional layers to generate the detection results. For each LRAU, we also change the dimension of the generated attention states because the model not only uses the attention state to predict the fine-grained classification scores, but also generates the bounding box adjustments. The information of the filters in LRAU for VDFR is shown in Table 4.2.

An overview of the proposed SSD-LRAU is illustrated in Figure 4.3. As illustrated in the figure, we add five LRAUs to receive feature maps of five scales. Then five attention

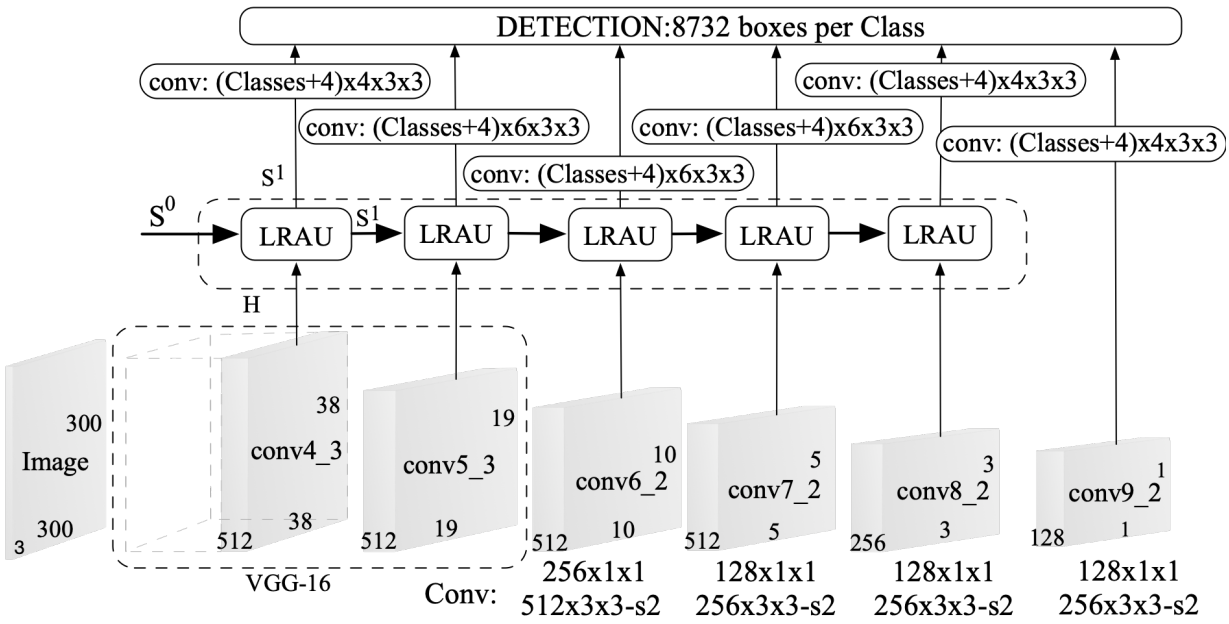


Figure 4.3: The framework of the SSD-LRAU. Five LRAUs are added to SSD to process five feature maps and generate five attention states. Five attention states and the last feature maps are used to generate the detection results.

states are generated by these LRAUs. Each attention state is sent to two convolutional layers. One convolutional layer is responsible for predicting $4 \times B \times h \times w$ bounding box adjustments, and another is used to predict $n \times B \times h \times w$ fine-grained category classification results, where B is the number of default anchor boxes of each layer, h and w is the size of the corresponding attention state (each attention state cell generates detection results for B boxes). The backbone of the SSD network is the truncated VGG16. We remove the $Conv6$ and $Conv7$ layers of SSD and directly use the feature maps generated by $Conv5\_3$ layer. The rest of the convolutional layers in our model remain the same as SSD. Also, we continue to use the same objective loss function as SSD.

During the training procedure, the default bounding boxes will match the corresponding ground truth box. This is determined by comparing the Jaccard overlap (IOU) between the ground truth box and each default anchor box. We match default anchor boxes to any ground truth box with IOU score $\geq$0.5.

We continue to use the same objective loss function as SSD. Let $p_{ij}^{n} = \{1, 0\}$ is the indicator for matching the i-th default anchors to the j-th ground truth box of category n. The objective function is the weighted sum of the classification (confidence) loss (cla) and the localization loss (loc):

$$L(p, y, b, t) = \frac{1}{N}(L_{cla}(p, y) + \alpha L_{loc}(p, b, t)), \tag{4.9}$$

where N is the number of matched default anchors. The loss is set to 0 when $N = 0$. The

classification loss $L_{cla}(p, y)$ is the Softmax loss:

$$L = -\sum_{i \in Pos}^{N} p_{ij}^{n} log\bar{y}_{i}^{n} - \sum_{i \in Neg} log\bar{y}_{i}^{0}, \tag{4.10}$$

where

$$\bar{y}_{i}^{n} = \frac{exp(y_{i}^{n})}{\sum_{n} exp(y_{i}^{n})}. \tag{4.11}$$

The localization loss $L_{loc}(p, b, t)$ is a Smooth L1 loss [125] between the predicted box (b) and the ground truth box (t). The predicted box is generated based the default anchor box (db). Let (cx, cy) denotes the center location of the box, the localization loss is defined as:

$$L_{loc}(x, b, t) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} p_{i,j}^{k} smooth_{L1}(b_{i}^{m} - \hat{t}_{j}^{m})$$

$$\hat{t}_{j}^{cy} = (t_{j}^{cy} - db_{i}^{cy})/db_{i}^{h}, \quad \hat{t}_{j}^{cx} = (t_{j}^{cx} - db_{i}^{cx})/db_{i}^{w}, \quad \hat{t}_{j}^{h} = log(\frac{t_{j}^{h}}{db_{i}^{h}}), \quad \hat{t}_{j}^{w} = log(\frac{t_{j}^{w}}{db_{i}^{w}}), \tag{4.12}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if \ |x| < 1, \\ |x| - 0.5 & otherwise. \end{cases} \tag{4.13}$$

4.2 Experiments

Our experiments are conducted on three benchmark VMMR datasets, Stanford Cars [157], CompCars [265], and NTOU-MMR [1]; and one benchmark object detection dataset, PASCAL VOC [117]. The detailed information of these datasets is introduced in the following. The important information of these datasets is concluded in Table 4.3. Sample images in these datasets are shown in Figure 4.4.

4.2.1 Datasets

Stanford Cars [157]: The Stanford Cars dataset is mainly used for VMMR. The dataset contains 16,185 images of 196 classes. Images are mainly collected from the website and are split into 8,144 training images and 8,041 testing images. Images have vehicles with different poses.

CompCars [265]: The CompCars dataset is a large dataset that has been widely used in various vehicle-related tasks. We use the web-nature subset of the CompCars dataset. The web-nature subset contains 30,955 images of 431 car categories. Images are collected from the website and have vehicles with different poses. The dataset is split into 16,016 training images and 14,939 testing images.

(a) Stanford Cars (b) CompCars

(c) NTOU-MMR (d) PASCAL VOC

Figure 4.4: Samples from (a) Stanford Cars dataset, (b) CompCars dataset and (c) NTOU-MMR dataset (d) PASCAL VOC dataset.

NTOU-MMR [1]: The NTOU-MMR dataset was collected under the Vision-based Intelligent Environment (VBIE) project [3] and can be accessed at [1]. Images are mainly collected by surveillance cameras under different weather conditions such as sunny, cloudy, and rainy. This dataset also includes images with vehicles that are partially occluded by other objects. Therefore, the dataset can be used to demonstrate the effectiveness of our models in different challenging environmental conditions. The dataset was originally split into a training set containing 2,846 images and a testing set containing 3,793 images. The number of vehicle models is 29.

However, we find some problems with the NTOU-MMR dataset: (1) There are some duplicate images (same image but with different names); (2) Some images are labeled with a wrong class name; (3) The partition strategy of splitting the training and testing set is unclear. The bias partition could result in misleading results [230]. To fix the problems mentioned above, we remove the duplicate images and correct the labels of the images. Then, we follow the partition strategy mentioned in [230], i.e., for each class of vehicle models, the related images are randomly split into the training set and testing set in a

Table 4.3: The statistics of four benchmark datasets. BBox stands for bounding box

Datasets	Category	Train	Test	BBox
Stanford Cars [157]	196	8,144	8,041	√
CompCars [265]	431	16,016	14,939	√
NTOU-MMR[1]	29	2,846	3,793	-
VOC2007[117]	20	5,011	4,952	√
VOC2012[117]	20	11,540	-	√

Table 4.4: The number of training images and testing images in NTOU-MMR dataset (ratio 8:2)

Make	Toyota					
Model	Altis	Camry	Vios	Wish	Yaris	Previa
#Train	963	553	547	302	243	49
#Test	241	139	137	76	61	13
Make	Toyota				Honda	
Model	Innova	Surf	Tercel	RAV4	CRV	Civic
#Train	36	66	144	62	495	264
#Test	9	17	37	16	124	67
Make	Honda	Nissan				
Model	FIT	March	Livna	Teana	Sentra	Cefiro
#Train	83	154	202	76	73	122
#Test	21	39	51	19	19	31
Make	Nissan		Mitsubishi			
Model	X-Trail	Tiida	Zinger	Outlander	Savrin	Lancer
#Train	110	207	12	61	42	62
#Test	28	52	4	16	11	16
Make	Suzuki	Ford				
Model	Solio	Liata	Escape	Mondeo	Tierra	Total
#Train	100	15	89	86	52	5,270
#Test	26	4	23	22	13	1,332
						6,602

ratio of 8:2. The number of images for each category after the clean and partition step is shown in Table 4.4. We repeat the partition strategy 10 times to construct 10 different train-test splits. Experiments are conducted on these 10 train-test splits, and we report the average results over the 10 train-test splits.

PASCAL VOC [117]: The PASCAL Visual Object Classes (VOC) challenge is a benchmark for object classification and detection. We use the *train/val* of VOC 2007 and VOC 2012 (16,551 images) as the training set, and the *test* of VOC2007 (4,952 images) as the testing set. We adopt the evaluation procedure of PASCAL VOC to evaluate the object detection performance of our model.

4.2.2 Implementation Details

Our models are implemented with PyTorch [206] and run on a workstation with a single GPU of GTX1080Ti. We pre-train ResNet50, VGG16 and InceptionV3 on the ImageNet

dataset [235], and then combine with LRAU fine-tuning on the target datasets. For the general setting of VMMR experiments, we trained our models with Stochastic Gradient Descent (SGD) with a weight decay of 0.0005 and a momentum of 0.9. We used images of size 224 $\times$ 224 as the input of our models. Data augmentations such as horizontal flip and random crop are applied to the original images. The convolutional weights of LRAU are initialized by using the method presented in [134], biases are initialized to zero.

When our models are trained on the Stanford and Compcars datasets, the initial learning rate is set to 0.01. After 40 epochs, the learning rate is reduced by a factor of 0.5 every 20 epochs. Models are trained for 120 epochs in total. The vehicle is cropped out from the image with the help of the bounding box in the last 40 epochs when models are trained on the CompCars dataset. When conducting experiments on the NTOU-MMR dataset, models are trained for 60 epochs. The initial learning rate is set to 0.01 and divided by 0.5 at the 30th, 40th, and 50th epoch. In addition, instead of using the pre-trained ResNet50, VGG16 and InceptionV3, the parameters of these models are randomly initialized.

The main metric for evaluating the VMMR model is accuracy, which is the ratio of the number of correctly classified images to the total number of test images. In addition, when conducting experiments on the NTOU-MMR dataset, we also report the class-wise accuracy, and average class-wise accuracy (ACC2) [230]:

$$ACC2 = \frac{\sum_i^n \frac{correct_i}{Total_i}}{n}, \tag{4.14}$$

where n is the number of vehicle categories, $Total_i$ is the number of testing images of class i, $correct_i$ is the number of correctly classified images of class i.

For the VDFR experiments, we still use SGD with the same setting mentioned above. We adopt the learning rate warming up strategy in [172] for the first 5 epochs (i.e., the learning rate is increased from 10^{-6} to 4×10^{-3}). After the first 5 training epochs, the initial learning rate is set to 0.004 and divided by 0.1 at 150, 180, and 210 epochs. The total number of training epochs is 220. The size of the input image is 300 $\times$ 300. The number of the default anchor boxes for each layer is shown in Figure 4.3. Other training strategies (such as hard negative mining, data augmentation, and settings of default anchors) are still the same as SSD. We use the mean Average Precision (mAP) [125] as the evaluation metric.

4.2.3 Ablation Study

In this subsection, we systematically evaluate the performance of the proposed three models under different parameter settings.

The number of attention masks and LRAUs. We investigate the influence of two parameters, namely the number of attention masks, and the number of LRAUs, on the performance of ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU. The number of attention masks k controls the number of attention masks generated in each LRAU, and k is set from 1 to 4. The number of LRAUs d controls the number of feature maps

Table 4.5: The influence of the number of LRAUs d, and the number of attention masks k for each LRAU on the Stanford Cars dataset.

	ResNet50-LRAU			VGG16-LRAU			InceptionV3-LRAU		
k	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$
$k=1$	93.57%	93.61%	93.52%	92.95%	93.11%	93.26%	92.30%	92.20%	91.79%
$k=2$	93.79%	**93.94%**	93.63%	93.20%	93.12%	93.10%	92.28%	92.51%	92.38%
$k=3$	93.55%	93.71%	93.68%	**93.51%**	93.17%	93.07%	**92.64%**	92.17%	92.53%
$k=4$	93.25%	93.60%	93.61%	93.16%	93.06%	92.97%	92.45%	92.18%	92.61%
baseline	ResNet50	91.78%		VGG16-GAP	91.23%		InceptionV3	90.95%	

Table 4.6: The influence of the number of LRAUs d, and the number of attention masks k for each LRAU on the CompCars dataset.

	ResNet50-LRAU			VGG16-LRAU			InceptionV3-LRAU		
k	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$
$k=1$	97.9%	98.18%	98.04%	97.78%	97.85%	97.92%	97.26%	97.38%	97.29%
$k=2$	98.11%	98.19%	98.18%	98.23%	98.18%	98.11%	**97.51%**	97.43%	97.44%
$k=3$	**98.25%**	98.19%	98.15%	**98.31%**	98.17%	98.29%	97.36%	97.46%	97.47%
$k=4$	98.19%	98.15%	98.02%	98.20%	98.00%	98.06%	97.44%	97.28%	97.42%
baseline	ResNet50	96.73%		VGG16-GAP	96.18%		InceptionV3	96.61%	

Table 4.7: The influence of the number of LRAUs d, and the number of attention masks k for each LRAU on the NTOU-MMR dataset.

	ResNet50-LRAU			VGG16-LRAU			InceptionV3-LRAU		
k	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$	$d=2$	$d=3$	$d=4$
$k=1$	98.44%	98.51%	98.36%	99.38%	99.39%	99.38%	98.43%	98.52%	98.55%
$k=2$	**98.66%**	98.53%	98.45%	99.40%	99.26%	99.39%	98.38%	98.46%	98.55%
$k=3$	98.53%	98.56%	98.08%	**99.41%**	99.32%	99.38%	**98.56%**	98.55%	98.34%
$k=4$	98.53%	98.61%	98.23%	99.38%	99.39%	99.38%	98.35%	98.53%	98.51%
baseline	ResNet50	98.00%		VGG16-GAP	97.53%		InceptionV3	96.32%	

that the model will combine, and it is determined by the structure of the standard CNN architecture. For example, as we mentioned earlier, ResNet has four convolutional layer groups, and different convolutional layer groups generate feature maps of different scales. Therefore, we can add four LRAUs to ResNet in total. In addition, we need to add at least two LRAUs so that features of different scales can be combined.

To find the optimal parameter settings for ResNet50-LRAU, InceptionV3-LRAU and VGG16-LRAU, and to study the changes in model performance on each dataset, we conduct experiments on Stanford Cars, CompCars, and NTOU-MMR. Table 4.5, 4.6, and 4.7 show the experimental results on different datasets using different k and d for ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU. We can observe that the recognition results of these three models are better than the corresponding baselines. Generally, the model recognition accuracy is improved when increasing k. For example, when $d=2$, the recognition accuracy of VGG16-LRAU is increased from 92.95% ($k=1$) to 93.51% ($k=3$) on the Stanford Cars dataset. This means that using the attention mask to extract discriminative part features can improve the recognition accuracy. Based on the experimental results, we can observe that the best results are usually obtained by setting k to 2 or 3. However, a large k (k greater than 3) can cause accuracy degradation. By observing these attention masks, we found that some attention masks focus on some unrelated regions, or the value of the whole mask is near 1. In addition, deploying two LRAUs to combine the

Table 4.8: The number of parameters (Params.) and floating point operations (FLOPs) of InceptionV3-LRAU, ResNet50-LRAU and VGG16-LRAU when evaluating on the Stanford Cars dataset

Model	Params. (10^6)	GFLOPs	Accuracy
ResNet152-CMP (r16) [186]	61.5	-	92.96%
ResNet50	23.9	4.12	91.78%
ResNet50-LRAU (d3k1)	24.8	4.27	93.61%
ResNet50-LRAU (d3k2)	25.2	4.30	**93.94**%
ResNet50-LRAU (d3k3)	25.8	4.35	93.71%
ResNet50-LRAU (d3k4)	26.6	4.41	93.60%
VGG16-CMP (r2) [186]	21.3	-	91.94%
VGG16	135.1	15.50	91.28%
VGG16-GAP	14.8	15.38	91.23%
VGG16-LRAU (d2k1)	15.1	15.43	92.95%
VGG16-LRAU (d2k2)	15.4	15.44	93.20%
VGG16-LRAU (d2k3)	15.8	15.45	**93.51**%
VGG16-LRAU (d2k4)	16.4	15.46	93.16%
InceptionV3	21.8	2.85	90.95%
InceptionV3-LRAU (d2k1)	23.5	2.89	92.30%
InceptionV3-LRAU (d2k2)	25.6	2.91	92.28%
InceptionV3-LRAU (d2k3)	29.1	2.96	**92.64**%
InceptionV3-LRAU (d2k4)	34.0	3.02	92.45%

Table 4.9: The number of parameters (Params.) and floating point operations (FLOPs) of InceptionV3-LRAU, ResNet50-LRAU and VGG16-LRAU when evaluating on the Comp-Cars dataset

Model	Params. (10^6)	GFLOPs	Accuracy
ResNet152-CMP (r8) [186]	64.8	-	97.01%
ResNet50	24.4	4.12	96.73%
ResNet50-LRAU (d2k1)	26.3	4.26	97.9%
ResNet50-LRAU (d2k2)	27.6	4.29	98.11%
ResNet50-LRAU (d2k3)	29.6	4.33	**98.25**%
ResNet50-LRAU (d2k4)	32.4	4.40	98.19%
VGG16-CMP (r2) [186]	21.4	-	97.80%
VGG16	136.0	15.50	96.15%
VGG16-GAP	14.9	15.38	96.18%
VGG16-LRAU (d2k1)	15.9	15.47	97.78%
VGG16-LRAU (d2k2)	17.3	15.50	98.23%
VGG16-LRAU (d2k3)	19.3	15.54	**98.31**%
VGG16-LRAU (d2k4)	22.1	15.61	98.20%
InceptionV3	22.7	2.85	96.61%
InceptionV3-LRAU (d2k1)	27.4	2.96	97.36%
InceptionV3-LRAU (d2k2)	37.6	3.08	**97.51**%
InceptionV3-LRAU (d2k3)	54.6	3.29	97.36%
InceptionV3-LRAU (d2k4)	78.2	3.59	97.44%

higher level features can achieve a better result, which indicates the benefit of fusing features of different scales. However, according to the experimental results, combining lower level features does not seem very helpful. One possible explanation is that the information carried by the lower layers is very fundamental (e.g., line, corner), and combining these

Table 4.10: The number of parameters (Params.) and floating point operations (FLOPs) of InceptionV3-LRAU, ResNet50-LRAU and VGG16-LRAU when evaluating on the NTOU-MMR dataset

Model	Params. (10^6)	GFLOPs	Accuracy
ResNet50	23.57	4.12	98.00%
ResNet50-LRAU (d2k1)	23.66	4.13	97.90%
ResNet50-LRAU (d2k2)	23.67	4.13	98.11%
ResNet50-LRAU (d2k3)	23.68	4.13	**98.25**%
ResNet50-LRAU (d2k4)	23.70	4.13	98.19%
VGG16	134.38	15.50	96.73%
VGG16-GAP	14.73	15.38	97.53%
VGG16-LRAU (d2k1)	14.77	15.41	97.78%
VGG16-LRAU (d2k2)	14.78	15.41	98.23%
VGG16-LRAU (d2k3)	14.79	15.41	**98.31**%
VGG16-LRAU (d2k4)	14.80	15.41	98.20%
InceptionV3	21.84	2.85	96.32%
InceptionV3-LRAU (d2k1)	21.95	2.85	98.43%
InceptionV3-LRAU (d2k2)	21.99	2.85	98.38%
InceptionV3-LRAU (d2k3)	22.07	2.85	**98.56**%
InceptionV3-LRAU (d2k4)	22.18	2.85	98.35%

fundamental features is not very useful to improve the model recognition performance. Instead, features of higher layers are more class-specific [292], and combining them can improve the model performance.

For VGG16-LRAU, the optimal recognition result on each dataset is achieved by setting $d = 2$ and $k = 3$. For ResNet50-LRAU, we observe that the parameter settings to achieve the optimal result on each dataset are different. For example, the best recognition result 93.94% on Stanford Cars is achieved by setting $d = 3$ and $k = 2$, and the optimal parameter setting on CompCars dataset is $d = 2$ and $k = 3$. For InceptionV3-LRAU, the optimal results on StanfordCars and NTOU-MMR dataset are achieved by setting $d = 2$ and $k = 3$, and the optimal parameter setting on CompCars is $d = 2$ and $k = 2$. By taking the computational complexity (to be shown in the next section) into consideration, we believe $d = 2$ and $k = 3$ are better choices for ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU, as they give the best trade-off between computational complexity and accuracy.

Computational Complexity. The number of parameters and floating point operations (FLOPs) [197] are usually used to measure the computational complexity of deep learning models. In Table 4.8, 4.9, and 4.10, we show the number of parameters and floating point operations (FLOPs) of each model when recognizing vehicle models on different datasets. For simplicity, we only show the data related to the optimal parameter setting for each model. We also show the number of parameters of ResNet152-CMP [186] and VGG16-CMP [186] with the optimal settings on each dataset.

In general, as d and k increase, the number of parameters and the number of FLOPs also increase. We think the increment is acceptable because the increment of model size and FLOPs is small, and the recognition accuracies of our models are significantly improved. For example, in Table 4.8, the number of parameters of ResNet50-LRAU (d3k2) is 25.2 $\times$

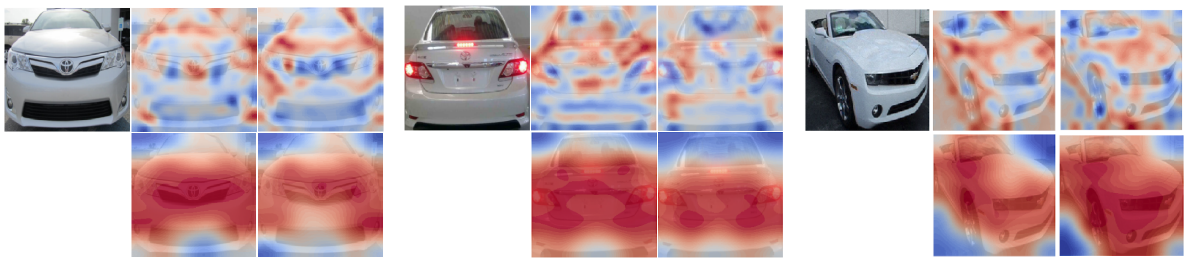
Table 4.11: The influence of the Global Average Pooling (GAP) layer and Global Max pooling (GMP) layer for ResNet50-LRAU and VGG16-LRAU. Experiments are conducted on the Stanford Cars dataset

Model	GAP	GMP
ResNet50-LRAU (d3k1)	93.38%	93.61%
ResNet50-LRAU (d3k2)	93.48%	**93.94**%
ResNet50-LRAU (d3k3)	93.53%	93.71%
ResNet50-LRAU (d3k4)	93.30%	93.60%
VGG16-LRAU (d2k1)	92.63%	92.95%
VGG16-LRAU (d2k2)	92.66%	93.20%
VGG16-LRAU (d2k3)	92.76%	**93.51**%
VGG16-LRAU (d2k4)	92.81%	93.16%

10^6, which is only a small percentage increase (5.4%) compared with ResNet50. We can observe that compared to ResNet152-CMP [186], ResNet50-LRAU (d3k2) can achieve a better recognition result with only 40.9% parameters. In addition, our models can achieve a high processing speed. For example, the average processing speeds of ResNet50-LRAU (d3k2), VGG16-LRAU (d2k3), and InceptionV3-LRAU (d2k3) are 138 FPS, 204 FPS, and 74 FPS (with a test batch size of 1). We also show the size of the original VGG and the VGG16-GAP. The size of VGG16-GAP is significantly reduced by removing the extra FC layer, and the accuracy remains almost the same. VGG16-LRAU (d2k3) obtains 2.23% relative improvement on accuracy compared with VGG16, and the number of parameters is reduced by 88.30%. InceptionV3-LRAU (d2k3) also obtains 1.69% accuracy improvement compared with InceptionV3, and the number of FLOPS is increased by 3.9%. We notice the number of parameters and FLOPs increments of InceptionV3-LRAU are larger than that of ResNet50-LRAU and VGG16-LRAU. As we mentioned before, we choose to use the convolutional filters with a larger kernel size to reduce the size of the attention state so that the features extracted by different units can be combined. The use of large kernel size filters leads to a greater increase in model computational complexity. However, LRAU can also be directly added (without changing the filter kernel size) to receive the feature maps generated by the last three Inception modules and obtain similar accuracy (92.41%).

In Table 4.9, we can observe that the number of parameters and FLOPs increases more because there are 431 vehicle models that need to be recognized in the Compcars dataset. The increment is still acceptable since the models can still achieve very high processing speed. For example, the average processing speeds of ResNet50-LRAU (d2k3), VGG16-LRAU (d2k3) and InceptionV3-LRAU (d2k2) are 135 FPS, 192 FPS and 68 FPS (with a batch size of 1). With the increment of k ($d = 2$), the number of FLOPs of ResNet50-LRAU are only increased by 0.14 GFLOPs ($k = 1$), 0.17 GFLOPs ($k = 2$), 0.21 GFLOPs ($k = 3$) and 0.28 GFLOPs ($k = 4$). Moreover, the number of FLOPs of VGG16-LRAUs with different k is less than that of VGG16, and VGG16-LRAU (d2k3) achieves the best recognition result on the Compcars dataset. In addition, in Table 4.10, we note that the model size and the number of FLOPs of ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU are almost unchanged when recognizing a small number of vehicle models, and the recognition accuracies of our models are significantly improved.

GMP vs. GAP. As we mentioned in Section 4.1.1, we use the GMP layer to process



(a) (b) (c)

Figure 4.5: Samples of attention masks. The upper row shows attention masks with an original size of 14×14. The bottom row shows attention masks with an original size of 7×7. We can observe that different attention masks highlight different discriminative regions of the image.

the attention state generated by the last LRAU. Another method to achieve this is to use the GAP layer. During training, the GMP layer only backpropagates the gradient to the location with the highest response value, while the GAP layer treats every location equally. Since LRAU is designed to selectively extract features, we consider the GMP layer to be a better choice for our model. Therefore, to check the influence of using the GAP layer and the GMP layer, we also conduct related experiments on the Stanford Cars dataset to check the changes in the model performance of ResNet50-LRAU and VGG16-LRAU. The experimental results are shown in Table 4.11. By using the GAP layer, we can observe that with the setting of $d = 3$ and $k = \{1, 4\}$, the accuracy of the ResNet50-LRAU is reduced by an average of 0.29%. Also, with the setting of $d = 2$ and $k = \{1, 4\}$, the accuracy of the VGG16-LRAU is reduced by an average of 0.49%. The experimental results demonstrate the GMP layer is a better choice for our model.

4.2.4 Visualization

The visualization of the attention masks is shown in Figure 4.5. The size of the attention masks is much smaller than the input image. Therefore, for better visualization, we resize (by interpolating) these attention masks and overlay them on the input image. To show the effectiveness of the attention mask at different viewpoints, we choose images with varying poses of vehicles. In Figure 4.5, the upper row shows the attention masks with an original size of 14×14. The bottom row shows the attention masks with an original size of 7×7. We can observe that these attention masks indicate some discriminative regions of the image, and each mask focuses on different regions. For example, for the left vehicle (Figure 4.5(a)), the left attention mask in the upper row pays more attention to the logo region, and the right attention mask focuses on the front edge of the vehicle.

To show the effectiveness of LRAU, we also visualize the feature maps generated by the standard CNN architecture and the attention state generated by the last LRAU in Figure 4.6. We use the same visualization method proposed in [273] to visualize the feature maps. By summing along the channel side of the feature maps, we can obtain a single-channel

(a) Original (b) Standard CNN (c) LRAU

Figure 4.6: Visualization of feature maps. (a) Original image. (b) The heatmaps of the last feature maps. (c) The heatmaps of the last attention state.

heatmap that highlights the prominent regions of the image. Figure 4.6(b) shows the heatmap generated according to the last feature maps generated by the standard CNN architecture. Figure 4.6(c) shows the heatmap generated based on the last attention state. We can observe that the heatmap in Figure 4.6(c) has a wider attention area than the heatmap in Figure 4.6(b). In addition, there are more discriminative regions shown in the heatmap of the attention state. For example, in Figure 4.6(c), the bottom heatmap highlights three discriminative regions (i.e., the left door, the left light, and the logo region). However, there is only one major discriminative region (near the left headlight of the vehicle), which is highlighted in the bottom heatmap in Figure 4.6(b).

Basically, for each LRAU, the attention mask is generated based on the features generated by the current layers and previous LRAU, which makes the attention mask show a wider attention region while highlighting the discriminative region. Then, the attention mask helps extract the discriminative features from the feature maps generated by the current convolutional layer. However, only using the extracted features may cause the loss of some keypoint features. Therefore, we combine the extracted features with the attention

Table 4.12: Comparison of the recognition results on the Stanford Cars, √ means testing with bounding box annotation, N/A indicates that bounding box annotation is not used

Type	Model	Test Anno.	Accuracy (%)
Previous	DVAN [284]	N/A	87.1
	WARN [222]	N/A	90.0
	BCNN (448) [168]	N/A	91.3
	M-CAN [290]	N/A	91.5
	RA-CNN (448) [120]	N/A	92.5
	MA-CNN (448) [287]	N/A	92.8
	BCNN-LSTM (448) [259]	N/A	93.4
	HBP (448)[270]	N/A	93.7
	ResNet50-SWP [141]	√	92.3
	PA-CNN [156]	√	92.8
	FCAN [179]	√	93.1
	ResNet101-SWP [141]	√	93.1
	DenseNet161-CMP [186]	√	**93.71**
Baseline	ResNet50	N/A	86.78
	VGG16-GAP	N/A	86.64
	InceptionV3	N/A	85.84
	ResNet50	√	91.78
	VGG16-GAP	√	91.23
	InceptionV3	√	90.95
Ours	ResNet50-LRAU (d3k2)	N/A	91.26
	VGG16-LRAU (d2k3)	N/A	92.13
	InceptionV3-LRAU (d2k3)	N/A	89.32
	ResNet50-LRAU (d3k2)	√	**93.94**
	VGG16-LRAU (d2k3)	√	93.51
	InceptionV3-LRAU (d2k3)	√	92.64

state generated by the previous LRAU to compensate for the loss. As shown above, the visualization of the last attention state shows more discriminative regions than the global feature maps. The final classification score is generated by evaluating these discriminative features and global features instead of only evaluating one global feature. The evaluation of two types of features provides more information to the model when recognizing a vehicle's fine-grained class.

4.2.5 Comparison Experiments on the Stanford Cars Dataset

Vehicle make and model recognition

Table 4.12 shows a comparison of the recognition results of our models with the recent state-of-the-art methods. In the table, *Previous* section shows the results cited from the original paper and results from *Baseline* and *Ours* section are derived from the experiments conducted based on the experimental setting we mentioned before. Without adopting the bounding box annotation at test time, the baseline VGG16-GAP achieves 86.64% accuracy. By combining two LRAUs, the proposed VGG16-LRAU (d2k3) achieves 92.13% recognition accuracy, with a significant improvement of 5.49%. VGG16-LRAU (d2k3) outperforms

Table 4.13: Comparison of vehicle detection and fine-grained recognition results on the Stanford Cars dataset. Param.: the number of parameters of the model. FPS: Frame Per Second

model	backbone	mAP (%)	Params. (10^6)	FPS
SSD	VGG	91.61	49.8	54
RFB	VGG	93.44	61.7	39
SSD-LRAU	VGG	**93.68**	**40.3**	**56**

WARN [222] by 2.13%. RA-CNN [120] and MA-CNN [287] crop out the part regions of the object according to their attention modules, and their models are trained with images of size 448×448. These part regions and large-scale images provide more detailed information, which allows their models to achieve higher results of 92.5% and 92.8%, respectively. HBP [270] is an improvement of BCNN; it combines multiple cross-layer bilinear features and conducts center cropping on the input image during testing, achieving an accuracy of 93.7%. Our ResNet50-LRAU (d3k2) achieves 93.94% accuracy by cropping the image with the help of the bounding box during testing. The best recognition accuracy, at 93.71% is achieved by DenseNet161-CMP [186]; ResNet50-LRAU outperforms it by a small margin, achieving the best recognition accuracy.

Vehicle detection and fine-grained recognition

We compare SSD-LRAU with two baselines SSD and RFB, as they both adopted VGG as the feature extraction network. Since there is no paper that reported the one-stage VDFR results, we re-implemented SSD and RFB and trained them with the same experimental setting. RFB net proposed RFB to simulate the human visual systems. The authors replaced the top convolutional layers of the SSD with RFB, which improved the feature extraction performance of the model. For the SSD-LRAU, the number of attention masks in each LRAU is fixed to 1. When we increase the number of attention masks, the model size increases significantly, but no obvious improvement in the model performance is found. The experimental result of VDFR is shown in Table 4.13. By combining five LRAUs, the mAP of SSD-LRAU is improved to 93.68%. Compared with SSD, which is 2.07% relative gain, RFB achieves 93.44% mAP; our model achieves a better result with only 65.3% parameters. Meanwhile, SSD-LRAU has the fastest processing speed (56FPS) among these three models. Figure 4.7 shows the sample images of the VDFR results of SSD-LRAU.

4.2.6 Comparison Experiments on the CompCars Dataset

Vehicle make and model recognition

The recognition results on CompCars dataset are summarized in Table 4.14. ResNet50, VGG16-GAP, and InceptionV3 achieve 96.73%, 96.18%, and 96.61% recognition accuracy. By combining multi-scale features, the performance of these baselines is improved, e.g., from 96.73% to 98.25% for ResNet50-LRAU. The improvement shows our models have

Figure 4.7: Sample images of the vehicle detection and fine-grained recognition result. Each image shows the predicted bounding box, the fine-grained label in a structure of make, model, type and year, and the confidence score.

Table 4.14: Comparison of the recognition results on the CompCars dataset

Type	Model	Accuracy (%)
Previous	AlexNet [265]	81.9
	OverFeat [265]	87.9
	Hu et al. [140]	91.0
	GoogLeNet [265]	91.2
	ResNet50-SWP [141]	97.5
	ResNet101-SWP [141]	97.6
	ResNet152-CMP [186]	97.01
	VGG-CMP [186]	97.80
	DenseNet161-CMP [186]	**97.89**
Baseline	ResNet50	96.73
	VGG16-GAP	96.18
	InceptionV3	96.61
Ours	ResNet50-LRAU (d2k3)	98.25
	VGG16-LRAU (d2k3)	**98.31**
	InceptionV3-LRAU (d2k2)	97.51

a stable performance in recognizing vehicle models. Hu et al. [140] proposed a two-stage framework to localize and recognize the vehicle. Their model requires bounding box annotation and an auxiliary dataset that contains saliency annotation to train the model. ResNet101-SWP [141] proposed a spatial weighted pooling layer to generate the attention mask to select the discriminative features, which achieves 97.6% accuracy. The recently proposed DenseNet161-CMP [186] achieves 97.89% accuracy on this dataset. VGG16-LRAU outperforms it by 0.42%, achieving the best recognition result on the CompCars dataset.

Table 4.15: Comparison of the vehicle detection and fine-grained recognition results on the CompCars dataset

model	backbone	mAP (%)	Params. (10^6)	FPS
SSD	VGG	95.82	81.2	33
RFB	VGG	97.17	95.3	22
SSD-LRAU	VGG	**98.09**	**65.2**	**35**

Table 4.16: Comparison of the recognition results on the NOUT-MMR dataset

Type	Model	Accuracy (%)	Speed (FPS)
Previous	LHS+Naive Bayes [209]	85.9	1.73
	FID-SRC-HDC [91]	91.1	0.46
	BoSURF-SD-SVM [230]	**94.84**	7.5
Baseline	ResNet50	98.00	163
	VGG16-GAP	97.53	243
	InceptionV3	96.32	78
Ours	ResNet50-LRAU (d2k2)	98.66	143
	VGG16-LRAU (d2k3)	**99.41**	208
	InceptionV3-LRAU (d2k3)	98.56	76

Vehicle detection and fine-grained recognition

The VDFR results of the models on the CompCars dataset are shown in Table 4.15. SSD achieves 95.82% mAP. By incorporating multiple LRAUs, we can obtain 2.27% relative improvement. SSD-LRAU achieves mAP of 98.09%, which is also better than RFB (97.17%), and SSD-LRAU only uses 68.4% parameters when compared with RFB. When these models need to distinguish a large number of vehicle models, SSD-LRAU still has the fastest real-time processing speed.

4.2.7 Comparison Experiments on the NTOU-MMR Dataset

As the traditional machine learning methods usually conduct experiments on the NTOU-MMR dataset, we can compare our models with them by conducting experiments on the NTOU-MMR dataset under a similar setting. In addition, when using the pre-trained models, we found it is easy for the baselines and our models to get a very high accuracy (almost 100%). Thus, it is difficult to observe the performance difference between the baseline and the proposed models. Therefore, instead of using the pre-trained ResNet, VGG, and InceptionV3, the model parameters are randomly initialized.

The comparison of the traditional machine learning VMMR models with our models is shown in Table 4.16. Overall, in terms of the recognition accuracy and the processing speed, the proposed ResNet50-LRAU, VGG16-LRAU, and InceptionV3-LRAU are superior to the traditional VMMR models. The BoSURF-SD-SVM [230] is an improvement of the SURF method. The authors first extracted the SURF features of the image and retained the domain features in a dictionary. Then, these features were embedded in the BoSURF histogram. Finally, SVM was used to classify these features. The BoSURF-SD-SVM model achieves 94.84% recognition accuracy with a processing speed of 7.5 FPS, which

Table 4.17: The class-wise accuray (%) of ResNet50-LRAU ($d2k2$) on the NTOU-MMR dataset. ACC2: average class-wise accuracy. Acc: accuracy

Make	Toyota					
Model	Altis	Camry	Vios	Wish	Yaris	Previa
Acc	99.67	98.92	99.49	99.34	99.84	99.23
Make	Toyota				Honda	
Model	Innova	Surf	Tercel	RAV4	CRV	Civic
Acc	100	97.06	95.67	97.50	99.43	99.70
Make	Honda	Nissan				
Model	FIT	March	Livna	Teana	Sentra	Cefiro
Acc	98.10	96.67	99.22	97.37	94.21	90.97
Make	Nissan		Mitsubishi			
Model	X-Trail	Tiida	Zinger	Outlander	Savrin	Lancer
Acc	100	99.42	90.00	98.12	95.46	99.38
Make	Suzuki	Ford				ACC2
Model	Solio	Liata	Escape	Mondeo	Tierra	97.36
Acc	100	90.00	98.26	94.09	96.15	Acc
						98.66

was the best traditional VMMR model. The three baselines ResNet50, VGG16-GAP, and InceptionV3 all outperform it with a clear margin of 3.16%, 2.69%, and 1.48%, respectively, and the processing speeds of ResNet50, VGG16-GAP, and InceptionV3 are significantly faster than the processing speed of BoSURF-SD-SVM. By adding LRAUs to ResNet50, VGG16-GAP, and InceptionV3, the performance of our models is improved. In particular, a significant improvement of 1.88% is obtained for VGG16-LRAU (d2k3), achieving the best recognition accuracy of 99.41% on the NTOU-MMR dataset.

The processing speeds of ResNet50-LRAU (d2k2), VGG16-LRAU (d2k3), and InceptionV3 (d2k3) are 143 FPS, 208 FPS, and 76 FPS. Compared with the baselines, the processing speed of the proposed models is decreased. Since the frame rate of the common surveillance cameras is 25-30 FPS [2], our models can still meet the processing speed required for real-time applications. In addition, the processing speed shown in Table 4.16 is the average speed of processing 1332 images with a batch size of 1. Under this setting, we note the processing speed of VGG16-LRAU (d2k3) is faster than that of ResNet50-LRAU (d2k2). When we use a larger batch size, ResNet50-LRAU (d2k2) is indeed faster than VGG16-LRAU (d2k3) as ResNet50-LRAU has fewer FLOPs. For example, when the batch size is set to 4, the processing speed of ResNet50-LRAU (d2k2) and VGG16 (d2k3) is 10.1 ms (millisecond) and 13.5 ms per batch. When the batch size is set to 8, the processing speed of ResNet50-LRAU (d2k2) and VGG16 (d2k3) is 16.3 ms and 25.7 ms per batch.

To systematically compare our models with the related traditional machine learning VMMR models, we also report the class-wise accuracy and average class-wise accuracy (ACC2) of ResNet50-LRAU (d2k2), VGG16 (d2k3), and InceptionV3-LRAU (d2k3) in Table 4.17, 4.18, and 4.19. The values of ACC2 of ResNet50-LRAU (d2k2), VGG16-LRAU (d2k3) and InceptionV3-LRAU (d2k3) are 97.36%, 99.19%, and 97.43%. We notice that the performance of ResNet50-LRAU (d2k2) is relatively low on several vehicle models. One reason for this could be the lack of sufficient training images to train a model with very deep

Table 4.18: The class-wise accuray (%) of VGG16-LRAU ($d2k3$) on the NTOU-MMR dataset. ACC2: average class-wise accuracy. Acc: accuracy

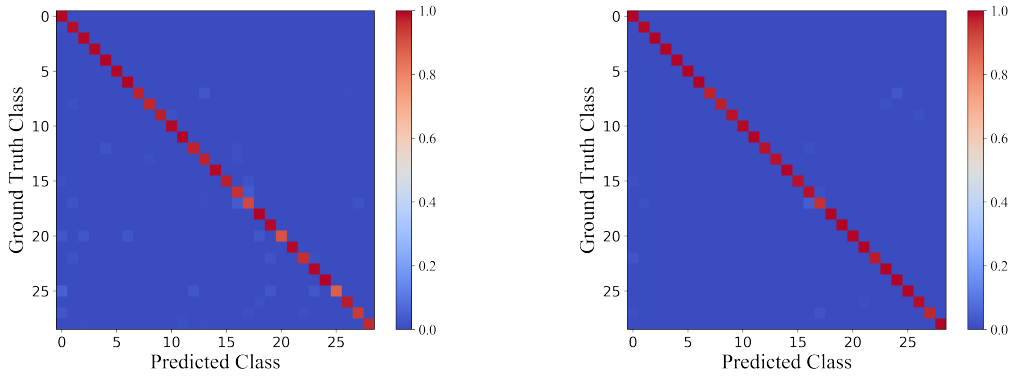
Make	Toyota					
Model	Altis	Camry	Vios	Wish	Yaris	Previa
Acc	99.84	99.42	99.78	99.74	99.67	100
Make	Toyota				Honda	
Model	Innova	Surf	Tercel	RAV4	CRV	Civic
Acc	100	97.65	97.84	99.38	99.84	99.85
Make	Honda	Nissan				
Model	FIT	March	Livna	Teana	Sentra	Cefiro
Acc	99.05	98.21	99.61	97.90	99.47	94.84
Make	Nissan		Mitsubishi			
Model	X-Trail	Tiida	Zinger	Outlander	Savrin	Lancer
Acc	100	100	100	100	99.09	100
Make	Suzuki	Ford				ACC2
Model	Solio	Liata	Escape	Mondeo	Tierra	99.19
Acc	100	100	100	95.45	100	Acc
						99.41

Table 4.19: The class-wise accuray (%) of InceptionV3-LRAU ($d2k3$) on the NTOU-MMR dataset. ACC2: average class-wise accuracy. Acc: accuracy

Make	Toyota					
Model	Altis	Camry	Vios	Wish	Yaris	Previa
Acc	99.59	99.06	99.34	99.47	99.84	98.46
Make	Toyota				Honda	
Model	Innova	Surf	Tercel	RAV4	CRV	Civic
Acc	98.89	94.71	96.76	98.12	99.52	99.40
Make	Honda	Nissan				
Model	FIT	March	Livna	Teana	Sentra	Cefiro
Acc	97.14	94.62	99.41	97.90	93.68	90.32
Make	Nissan		Mitsubishi			
Model	X-Trail	Tiida	Zinger	Outlander	Savrin	Lancer
Acc	100	98.85	97.5	99.38	92.73	99.38
Make	Suzuki	Ford				ACC2
Model	Solio	Liata	Escape	Mondeo	Tierra	97.43
Acc	99.62	92.50	99.13	93.18	96.92	Acc
						98.56

layers (50 layers for ResNet50), so the model does not acquire a good generalization ability. For example, there are only 15 training images for Ford Liata and 12 for Mitsubishi Zinger, and the recognition accuracy of ResNet50-LRAU (d2k2) on both vehicle models is only 90.00%. Surprisingly, VGG16-LRAU (d2k3) has a stable performance when recognizing different vehicle models, even though some vehicle models have a small number of training images. For some vehicle models, VGG16-LRAU (d2k3) even achieves 100% accuracy, such as with the Nissan X-Trail and Ford Liata. The ACC2 of BoSURF-SD-SVM is 86.78%, VGG16-LRAU (d2k3) outperforms it by a large margin of 12.43%.

Since ResNet50-LRAU (d2k2) and VGG16-LRAU (d2k3) have higher performance, we also show their confusion matrices to compare with the traditional machine learning



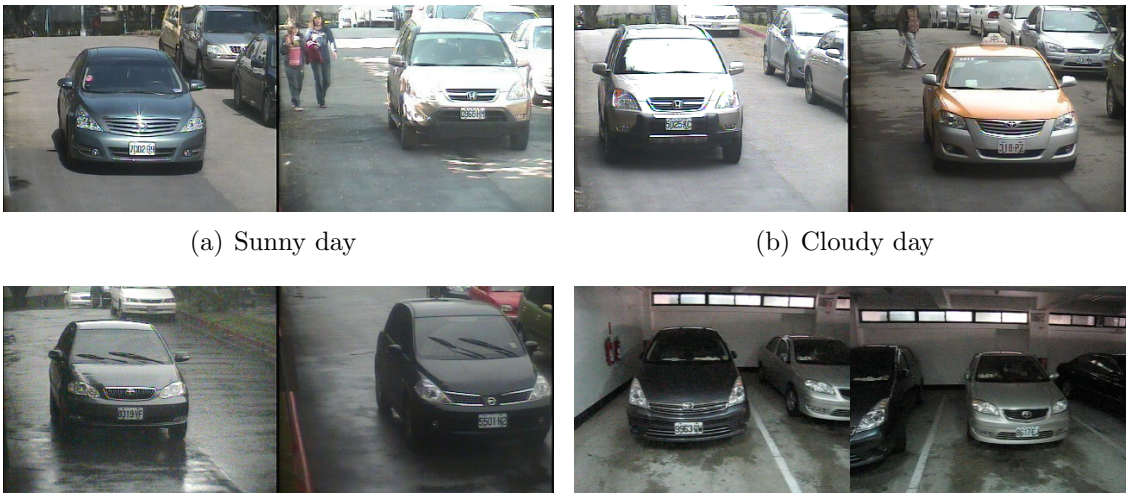
(a) ResNet50-LRAU (d2k2) (b) VGG16-LRAU (d2k3)

Figure 4.8: Confusion matrices for (a) ResNet50-LRAU (d2k2) and (b) VGG16-LRAU (d2k3). The result is the average of experiments on ten datasets.

method. Their confusion matrices on recognizing the 29 vehicle models in the NTOU-MMR dataset are shown in Figure 4.8. Siddiqui et al. [230] found most of the inaccurate predictions of their models are towards Class Toyota Altis and Toyota Camry. They thought this was mainly caused by the imbalanced number of training images for different vehicle models. For example, the number of training images of the Toyota Altis and Toyota Camry is largely greater than other vehicle models. For our proposed model, we did not observe the wrong predictions clearly towards any specific vehicle model, which shows our models have a robust performance in recognizing different models, even though the number of training images is imbalanced in the dataset. For ResNet50-LRAU (d2k2), we observe it would mistakenly recognize class 16 (Nissan Sentra) as class 17 (Nissan Cefiro) or vice-versa. By checking the images, we found the vehicles of these two classes are very similar, and the training images of both models are very few. We think ResNet50-LRAU (d2k2) has a poor performance in recognizing these two vehicle models, mainly due to the lack of training images. For VGG16-LRAU (d2k3), the recognition performance of recognizing these two vehicle models is better than ResNet50-LRAU (d2k2). We can also observe such an improvement in Table 4.17 and 4.18.

4.2.8 Performance Under Different Environmental Conditions

In order to evaluate the performance of our models under different environmental conditions, we manually marked the environmental conditions of each image in the NTOU-MMR dataset. There are four environmental conditions: (1) Sunny; (2) Cloudy; (3) Rainy; (4) Underground (parking lot). Vehicle images taken on a sunny day are marked as sunny. Sometimes, it is difficult to distinguish between sunny and cloudy days. The standard we use is to mark the image as sunny if it shows the obvious shadow of the objects or sunlight. When there is no clear shadow or rain, we mark the image as cloudy. If the vehicle is parked in an underground parking lot, we will mark the image as underground. In Figure 4.9, we show sample images under different weather conditions.



(a) Sunny day

(b) Cloudy day

(c) Rainy day

(d) Underground parking lot

Figure 4.9: Sample images under different environmental conditions

Table 4.20: Performance of ResNet50-LRAU ($d2k2$) under different environmental conditions

	Sunny	Cloudy	Rainy	Underground	Total
Total	418.6	648.3	179.3	85.8	1332
#Correct	410	640.6	177.8	85.8	1314.2
Accuracy	97.94	98.81	99.17	100	98.66

Table 4.21: Performance of VGG16-LRAU ($d2k3$) under different environmental conditions

	Sunny	Cloudy	Rainy	Underground	Total
Total	418.6	648.3	179.3	85.8	1332
#Correct	414.9	644.7	178.8	85.8	1324.2
Accuracy	99.11	99.44	99.72	100	99.41

Images taken in different environmental conditions can affect model performance. On sunny days, the strong sunlight can cause the vehicle's surface to reflect. Some keypoint regions may be difficult to identify. For example, it is difficult to recognize the logo of the vehicle in the first image in Figure 4.9(a). On rainy days, rain and windshield wipers could also affect the appearance of the vehicle. For vehicles parked in an underground parking lot, dim lights and a short distance from the nearby vehicles can also interfere with vehicle recognition.

In Table 4.20 and Table 4.21, we show the performance of ResNet50-LRAU (d2k2) and VGG16-LRAU (d2k3) under different environmental conditions. We observe that the recognition accuracy of ResNet50-LRAU (d2k2) on sunny days is lower than other environmental conditions. By checking the images of wrong predictions, we found that the reflection of the vehicle surface caused by sunlight would affect the model. However, VGG16-LRAU (d2k3) still achieves an excellent performance under this challenging environmental condition. The recognition accuracy on sunny days is improved to 99.11%.

Table 4.22: Comparison of the object detection results on the PASCAL VOC 2007 test set

	Model	mAP(%)	Params.(10^6)	FPS
Previous	YOLO (448) [213]	63.4	-	45
	YOLO VGG (448) [213]	66.4	-	21
	YOLOv2 (352) [214]	73.7	-	81
	SSD [173]	77.2	-	59
	RFB [172]	**80.5**	34.5	83
Our	SSD	77.6	26.3	**83**
	RFB	**80.0**	36.5	56
	SSD-LRAU	77.8	**21.6**	**83**

According to the high recognition accuracy on rainy days, we believe that the blurring effect of rainwater has little effect on our models. The experimental results demonstrate VGG16-LRAU (d2k3) has a stable performance under different challenging environmental conditions.

4.2.9 Object Detection

Even though SSD-LRAU is not designed for the object detection task, we also test the object detection performance of SSD-LRAU. We report the results from the original papers, and re-implement SSD and RFB for a fair comparison. These models are trained on the VOC2007+2012 training set and tested on the VOC2007 testing set. The comparison of these models is shown in Table 4.22. SSD-LRAU achieves 77.8% mAP, which is a comparable result to SSD. The VGG version of YOLO achieves 66.4% mAP, SSD-LRAU outperforms it with a large margin (11.4%). Moreover, SSD-LRAU is much smaller than YOLO, SSD, and RFB, and it has a faster detection speed. The results show that SSD-LRAU is also an excellent object detection model.

4.3 Summary

In this chapter, we proposed LRAU, a lightweight attention module designed to enhance the fine-grained feature extraction ability of standard CNN architectures. The proposed LRAU can help the model effectively extract discriminative features. By adding LRAU to two popular standard CNN architectures, we proposed two VMMR models, ResNet50-LRAU and VGG16-LRAU. We conducted a wide range of experiments on three benchmark VMMR datasets, including evaluation under different environmental conditions. The experimental results on these three challenging vehicle datasets demonstrate that our models surpass the traditional VMMR models and the state-of-the-art deep learning-based VMMR models. In particular, our models have stable performance under different environmental conditions.

In addition, we proposed a one-stage VDFR model SSD-LRAU to simultaneously detect and recognize the vehicle information. Our model achieves excellent fine-grained recognition performance and can be used in a real-time environment. We also conducted exper-

iments on an object detection dataset, and the results demonstrated that SSD-LRAU is also an excellent object detection model.

Chapter 5

Compact Attention Unit

Vehicle Re-identification (VReID) is an essential topic in the Intelligent Transportation System (ITS) and Smart City. As shown in Figure 5.1, given a query vehicle image, the goal of the vision-based VReID is to find the target vehicle from a large image database.

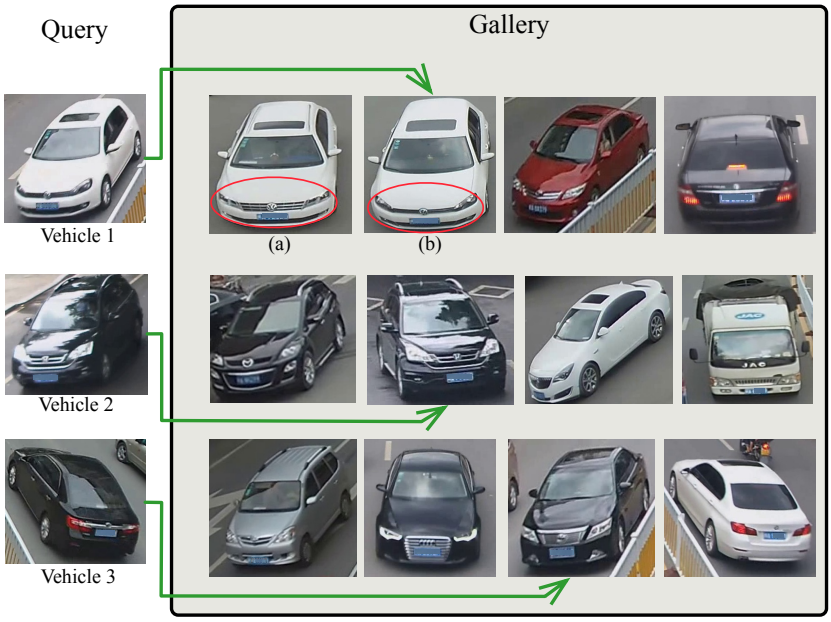


Figure 5.1: Vehicle re-identification (VReID) task. For each image in the query set, the goal of VReID task is to find the target vehicle from the gallery set. The query *vehicle 1* has a similar shape as *vehicle (a)* and *vehicle (b)*. To distinguish these two vehicles, we need to pay more attention to the discriminative regions (e.g., red circle region) and compare them.

In this chapter, we propose the Compact Attention Unit (CAU). By adding multiple CAUs to a truncated ResNet, we construct an efficient VReID model. The model extracts both the global feature and discriminative local feature and uses them to identify the target vehicle. More importantly, the model has a small model size and fast image processing

speed, making the model more suitable for smaller devices and the real-time environment. The content of this chapter has been published in *An Efficient Real-Time Vehicle Re-Identification Scheme Using Urban Surveillance Videos*.

5.1 The Proposed Approach

This section introduces the proposed attention module and the combination with standard CNN architectures. Generally, humans recognize a vehicle by checking its global appearance (the whole vehicle). For very similar vehicles, they pay extra attention to the local feature (e.g., headlight, taillight, logo). To make the model pay more attention to the local feature, we propose our attention model.

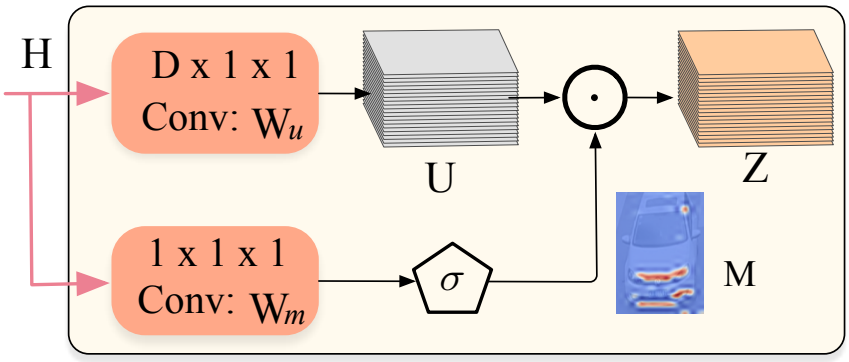


Figure 5.2: The structure of Compact Attention Unit. *Conv* denotes the convolutional layer.

5.1.1 Attention Module

Given an image x as input, the backbone network will generate a set of feature maps. The CAU takes the feature maps as input. Then, CAU will automatically locate the discriminative regions and generate attention features. The structure of the CAU is shown in Figure 5.2. In CAU, the input feature maps will first be processed by the convolutional layers with 1 $\times$1 kernel size. Let H represents the input feature maps, and W_u the overall parameters. The procedure is denoted by:

$$U = W_u * H, \tag{5.1}$$

where $U \in \mathbb{R}^{d \times h \times w}$ is the generated feature maps, h and w represent the height and width of the feature maps, respectively, d is the number of channels. The whole convolution operations are denoted by '*'. Then, a Batch Normalization (BN) layer [145] is used to normalize U.

As each channel of the feature maps can reflect some discriminative regions [287], a general way of generating the attention map is summing up (or averaging) the feature map

along the channel dimension to generate a heatmap. Then the most important region will have a higher value in the heatmap. By setting a threshold, the discriminative region can be explicitly selected (e.g., [249]). Instead, we adopt another 1 ×1 convolutional layer to produce the attention map directly:

$$M = \sigma(W_m * H), \tag{5.2}$$

where $M \in \mathbb{R}^{h \times w}$ is the attention map, which is a 2D matrix. The sigmoid σ function is used to normalize the value of the attention map between 0 and 1. The value of a location in the attention map is the weighted sum of all channels of the input feature maps in that location. Therefore, the filter can learn to give a high weight to the important channel and a low weight to the less important channel. Then, the discriminative regions will be shown in the generated attention map. Figure 5.2 shows the visualization of M, where the red regions indicate the discriminative parts of the vehicle.

Then, the attention map is used to extract the discriminative features from U.

$$Z_i(x, y) = M(x, y) \cdot U_i(x, y), \tag{5.3}$$

where Z_i represents the ith channel of the attention features, $x \in \{0, w\}$, $y \in \{0, h\}$. By multiplying the attention map with the feature maps, the value of the discriminative regions of all channels in the feature maps will be given a high weight. The size of the generated attention features is the same as the size of the input feature maps.

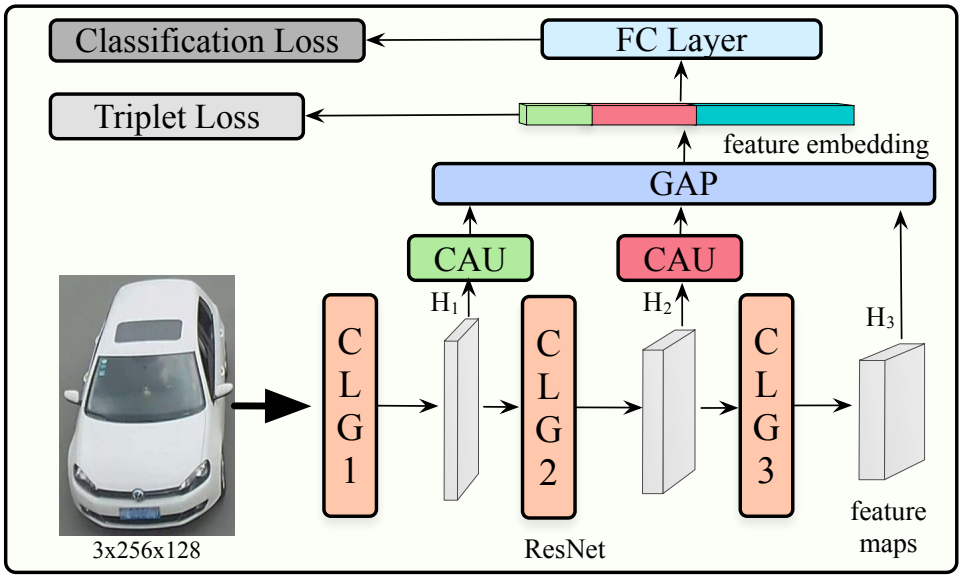


Figure 5.3: The framework of the ResNetT-CAU.

5.1.2 Combining CAU with ResNet

The proposed CAU can be added to most standard CNN architectures to extract the local features. We choose ResNet50 as the backbone feature extraction network. ResNet50

mainly consists of four Convolutional Layer Groups (CLGs). Each CLG consists of a different number of Bottleneck Blocks (convolutional layers). The numbers of channels of the output feature maps of these CLGs are 256, 512, 1024 and 2048, respectively. By checking the number of parameters of each CLG, we found the fourth CLG accounts for the largest proportion among the four groups. Also, the size of the images in the VReID datasets is smaller than the general input size of ResNet (224×224). Therefore, we think the fourth CLG of ResNet maybe redundant when used to address the VReID task. To construct a model with small model size, we remove the fourth CLG of ResNet50. The generated model is called ResNetT.

We add two CAUs to the end of the first and second CLG to receive the feature maps with 256 and 512 channels, respectively. The proposed model is called ResNetT-CAU. The structure is shown in Figure 5.3. As shown in Figure 5.3, these two CAUs extract the discriminative local features from the feature maps of different scales. The local and global features (generated by the last CLG) are processed by the Global Average Pooling (GAP) layer. After the pooling operation, tree vectors of sizes 1024, 512, and 256 are generated. The vectors are then concatenated together and normalized by the BN layer to generate a 1792-dimension feature embedding. The feature embedding is used to calculate the feature distance during testing.

5.1.3 Loss Function

We use the triplet loss function and the Cross-Entropy Loss (CEL) function to train our model. Given tree images (triplet) $\{q, x^p, x^n\}$ as inputs, where q (query) is the image of a specific vehicle, x^p (positive) is an image that contains a vehicle the same as q. In contrast, x^n is the negative sample of a vehicle different than q. Let $f(x)$ be the feature embedding of the input image x. The triplet loss function is defined as:

$$L_{tri} = \sum_{i=1}^{N} max\{0, D(f(q_i), f(x^p)) + \alpha - D(f(q_i), f(x^n))\}, \tag{5.4}$$

where N is the number of the samples, α is the minimum margin between the positive sample and negative sample. $D(a, b)$ calculates the feature distance of a, b. In general, the feature distance is the Euclidian distance. The function's goal is to make the feature embeddings of the images belonging to the same vehicle have a smaller feature distance, and make the feature embeddings of different vehicles have a large feature distance.

During testing, the feature embedding is directly used to compare the feature distance. During training, a Fully-Connected (FC) layer is used to process the feature embedding to generate the vehicle model classification score. The classification score is then further normalized by the softmax function. With the vehicle model classification loss function, vehicles of the same vehicle model will be clustered together, which will further improve the model performance. In our work, we adopt the CEL function as the classification loss function:

$$L_{cel} = -\sum_{i=1}^{c} \hat{y}_i log y_i, \tag{5.5}$$

where y_i is the predicted probability for class i, c is the number of vehicle model categories, $\hat{y}$ is the one-hot encoded ground-truth label vector. However, by simply using the CEL function, the model will cluster vehicles of the same model tightly, which will affect the model performance when recognizing two different vehicles of the same model. Therefore, we adopt the label smooth strategy [199]: $\hat{y}$ is redefined as $\hat{y}_i(i = j) = 1 - \delta$ and $\hat{y}_i(i \neq j) = \frac{\delta}{c-1}$, where j is the target category. δ is set to 0.1 in our experiments.

5.2 Experiments

5.2.1 Dataset and Evaluation Metrics

VeRi [175]: The images in this dataset are captured by 20 surveillance cameras deployed along a circular road. Each vehicle is captured by 2-18 cameras with different viewpoints under different environmental conditions. The dataset contains a training set and a testing set. The training set contains 37,781 images of 576 vehicles. The testing set is further split into a gallery set containing 11,579 images of 200 vehicles and a query set containing 1,678 images of 200 vehicles.

VehicleID [170]: The entire dataset contains 221,567 images of 26,328 vehicles, which are also captured by several surveillance cameras. The images are only captured from the front or the back viewpoint. The training set contains 110,178 images of 13,134 vehicles. VehicleID provides several testing sets. We mainly use the three commonly used test sets to test our models, namely Test800, Test1600, Test2400. Test800 contains 5,693 query images and 800 gallery images of 800 vehicles. Test1600 contains 11,777 query images and 1,600 gallery images of 1,600 vehicles. Test2400 contains 17,377 query images and 2,400 gallery images of 2,400 vehicles.

Evaluation Metrics. For the performance evaluation, we follow the evaluation procedures proposed in [175, 170]. We use the mean Average Precision (mAP) and the rank-1 and rank-5 identification rate to evaluate the model performance. The two metrics are widely used in the re-identification field. The mAP reflects the average precision of the model for all queries. The rank-k identification rate represents the percentage that the correct matches appear in the top-k results. We use the cosine distance and adopt Deep Spatial Reconstruction (DSR) [135] as the matching method.

5.2.2 Implementation Details

The models are implemented with PyTorch [206]. Experiments are conducted on a workstation with one GTX1080Ti GPU. The size of the input image of the model is 256 $\times$ 128. We use the horizontal flip and Random Erase [291] methods to augment the training image.

The model is trained for 120 epochs in total. The initial learning rate is 0.00035, and the learning rate is reduced by a factor of 0.1 (learning rate decay) at 40th and 90th epoch.

The training batch size is 64. We use the pre-trained ResNet as the backbone network, which is first trained on the ImageNet dataset [235], and then combined with CAU fine-tuned on the target datasets. The filter weights of CAU are initialized using the method introduced in [134], and biases are initialized to zero. The vehicle ID and vehicle model annotations are used as the ground truth labels in training.

Learning rate warmup. We adopt a learning rate warmup method proposed in [135] for the first 10 training epochs. During each training iteration (train a batch of images), the learning rate changes based on the following equation:

$$lr = lr_{init} * wf, \tag{5.6}$$

$$wf = wf_{init} * (1 - \beta) + \beta, \tag{5.7}$$

where lr_{init} is the initial learning rate, wf is the warmup factor, which is initialized to 0.01. $\beta = \frac{iterations}{total\_warmup\_iterations}$, β is a factor used to adjust wf during each training iteration. In our experiment, the number of warmup epochs is set to 10, which is 5,890 warmup iterations for the VeRi dataset and 17,710 warmup iterations for the VehicleID dataset. Generally, at the beginning of the warmup iterations, the learning rate will be a value lower than the initial learning rate (0.00035). Then, the learning rate will be gradually increased to 0.00035. The learning rate warmup strategy can effectively prevent the over-fitting of batch training at the initial training stage and improve the model performance.

Hard triplet mining. For the triplet loss function, the selection of triplets is important. To make the training more efficient, during training, the inputs are a batch of images, and each image is taken as the query. For each vehicle in a training batch, we sample k images. For example, we select 16 vehicles in a training batch, each containing 4 images (samples). Therefore, the number of images belonging to a vehicle in a training batch is balanced. During training, for each vehicle, instead of randomly choosing the positive and negative samples, we adopt the strategy proposed in [138]. The positive sample is the one that has the largest distance with query among all positive samples, and the negative sample is the one that has the smallest distance with query among all negative samples.

5.2.3 Ablation Study

This section conducts a series of comparison experiments on the VeRi dataset to validate the effectiveness of our proposed VReID model. We propose three models that have a similar structure to our model. The structure of these three models is shown in Figure 5.4. Figure 5.4(a) is named ResNetT-Plain. In this model, we directly concatenate the features extracted by CLG3, CLG2, and CLG1. We can verify whether the model performance mainly comes directly from using feature maps of different scales. In Figure 5.4(b), instead of directly using the feature maps, we add extra convolutional layers to further extract the features. We named this model ResNetT-Conv. The model can help us verify the effectiveness of our CAU. In Figure 5.4(c), we add one FC layer to further process the concatenated feature embedding. We named this model ResNetT-CAUFC.

To systematically evaluate the proposed models, we also report the number of parameters, floating point operations (FLOPs) [197], and the processing time per batch (the

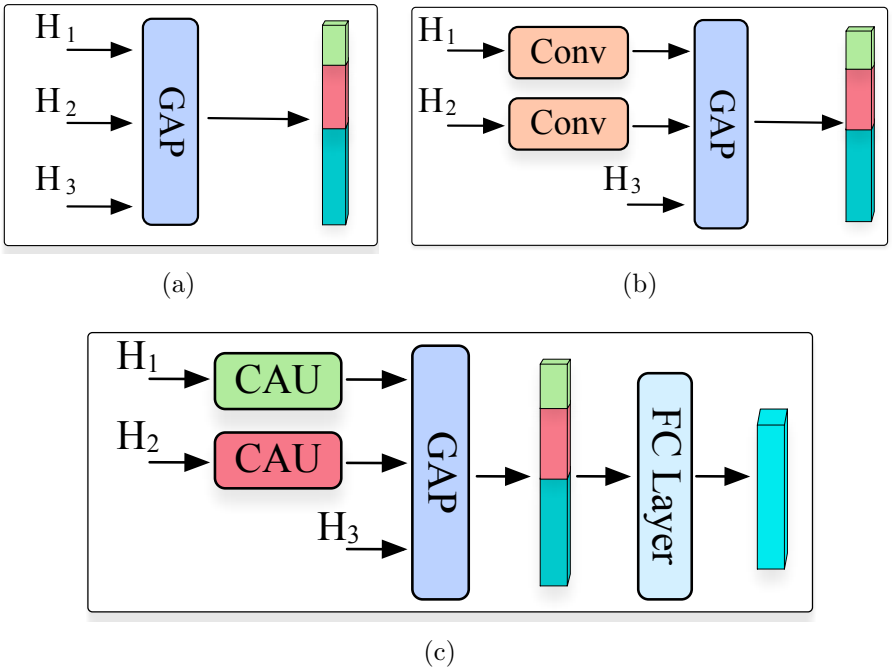


Figure 5.4: The structure of the models used for ablation study

Table 5.1: Ablation study results on the VeRi dataset (%). WU: Learning rate warmup. HT: Hard Triplet Mining

Model	P	F	T	mAP	Rank-1	Rank-5
ResNet(no WU)	24.69	4.06	5.5	70.77	92.67	96.84
ResNet(no HT)	24.69	4.06	5.5	73.03	93.44	96.96
ResNet	24.69	4.06	5.5	73.33	93.80	97.50
ResNetT(no WU)	9.13	2.15	4.6	63.87	89.7	96.00
ResNetT(no HT)	9.13	2.15	4.6	70.38	93.03	97.38
ResNetT	9.13	2.15	4.6	71.16	91.66	97.20
ResNetT-plain	9.57	2.15	4.6	71.37	92.25	97.02
ResNetT-Conv	9.9	2.42	5.1	74.19	94.10	97.20
ResNetT-CAU	9.91	2.42	5.1	**75.86**	94.87	**97.74**
ResNetT-CAUFC	11.3	2.42	5.2	75.38	**94.99**	97.56

Abbreviations: P: The number of parameters ($\times 10^6$). F: The number of FLOPs ($\times 10^9$). T: Processing time per batch (ms).

testing batch size is 128) of each model. These factors are usually used to evaluate the deep learning models. The evaluation results are shown in Table 5.1. In the table, we can observe that the learning rate warmup strategy and hard triplet mining can effectively improve the model performance, and they are beneficial for the truncated ResNet. For example, these two training strategies can improve the mAP of ResNetT by 7.29% and 0.78%, respectively.

When we remove the last CLG of ResNet, the mAP is decreased from 73.33% (ResNet) to 71.16% (ResNetT). We can also observe that the combination of feature maps of different scales (ResNetT-plain) can slightly improve the model performance. By further extracting feature maps, ResNetT-Conv achieves mAP of 74.19%, which is an excellent improvement. By adding CAUs to ResNet, the mAP is further improved to 75.86%. Compared with ResNetT, the mAP is improved by 4.7%, the rank-1 and rank-k identification rate is improved by 3.21%, and 0.54%, respectively. The results indicate that CAU can significantly improve the model performance. In addition, the mAP of ResNetT-CAU is 2.53% higher than that of ResNet, and the model size of ResNetT-CAU is only 40.14% of the original ResNet. The processing speed of ResNetT-CAU is 5.1ms per batch (one batch contains 128 images). As a comparison, the image processing speed of QD-DLF [296] is 11.2ms per image. ResNetT-CAU is significantly faster than QD-DLF, and our model has a better performance (shown in Section 5.2.4). The FC layer is often used to further process the concatenated feature embedding. Therefore, we add one FC layer to project the feature embedding to a 1024-dimensional feature embedding. We can observe the rank-1 identification rate is slightly improved.

Table 5.2: Comparison with the state-of-the-art VReID methods on the VeRi dataset (%)

Model	mAP	Rank-1	Rank-5
FACT [175]	19.92	59.65	75.27
SDC-CNN [297]	53.45	83.49	92.55
PROVID [176]	53.42	81.56	95.11
VAMI [294]	61.32	85.92	91.84
RAM [177]	61.5	88.6	94.0
QD-DLF [296]	61.83	88.50	94.46
VANet [94]	66.34	89.78	95.99
PAMTRI [247]	71.88	92.86	97.97
SAN [210]	72.5	93.3	97.1
He et al. [133]	**74.3**	**94.3**	**98.7**
ResNetT-CAU	**75.86**	**94.87**	97.74

5.2.4 Comparison with the State-of-the-Art Methods

Performance on VeRi

Table 5.2 shows the comparison results of our proposed model with the recent state-of-the-art methods on the VeRi dataset. Compared with the global feature extraction models (FACT [175], SDC-CNN [297]), ResNetT-CAU shows a significant model performance improvement. From the table, we can observe that the models ([210, 133, 177]) that using

Table 5.3: Comparison with the state-of-the-art VReID methods on the VehicleID dataset (%). R-1: rank-1; R-5: rank-5

Model	Test800		Test1600		Test2400	
	R-1	R-5	R-1	R-5	R-1	R-5
DRDL [170]	48.91	66.71	46.36	64.38	40.97	69.02
FACT [175]	49.53	67.96	44.63	64.19	39.91	60.49
VAMI [294]	63.12	83.25	52.87	75.12	47.34	70.29
SDC-CNN [297]	56.98	86.90	50.57	80.05	42.92	73.44
QD-DLF [296]	72.32	92.48	70.66	88.90	64.14	83.37
RAM [177]	75.2	91.5	72.3	67.7	67.7	84.50
GS-TRE [28]	75.9	84.2	74.8	83.6	74.0	82.7
ResNetT-CAU	**80.40**	**93.75**	**76.27**	**90.14**	**74.45**	**87.87**

both the global and local features achieve a better result than the models only use the global features. He et al. [133] adopted a detection network to locate the part regions, the model achieves mAP of 74.3%. Our model implicitly locates the discriminative regions without using any part annotation, achieving 75.86% mAP, which is the best result.

Performance on VehicleID

Table 5.3 shows the performance comparison of ResNetT-CAU with the state-of-the-art methods on the VehicleID dataset. We can observe that our model consistently achieves very high rank-1 and rank-5 identification rates on these three test sets. The results show that the performance of ResNetT-CAU on the VehicleID dataset is better than the recent state-of-the-art methods.

5.3 Summary

In this chapter, we proposed a Compact Attention Unit (CAU), which can effectively extract the discriminative local features. The proposed ResNetT-CAU retrieves and identifies vehicles by comparing the extracted global and discriminative local features. Extensive experiments on the VeRi and VehicleID dataset indicate the proposed ResNetT-CAU achieve the highest re-identification results on both datasets. Moreover, ResNetT-CAU has a small model size and a very fast processing speed (5.1 ms per batch), which allows it to be used in devices with poor hardware specifications, and still achieves excellent performance. For future work, we plan to add CAU on other CNNs to explore its compatibility.

Chapter 6

Conclusion and Future Work

In this section, we conclude the proposed vehicle recognition models and provide directions for future work.

6.1 Conclusion

This thesis proposes deep learning-based models for robust and real-time vehicle recognition. To improve the VMMR performance of standard CNN architectures, we proposed RAU. RAU effectively extracts the discriminative features of an object by evaluating features extracted from different layers. The efficiency of the proposed ResNet101-RAU is tested on three challenging VMMR benchmark datasets, i.e., Stanford Cars, CompCars, and CompCars Surveillance.

To construct an efficient real-time VMMR model, we simplified the RAU structure and proposed LRAU. We combine LRAU with ResNet50 and VGG16 to enhance the feature extraction ability of these two standard CNN architectures. Experimental results show that our models achieve state-of-the-art results on the Stanford Cars, CompCars, and NTOU-MMR datasets. Also, the performance improvements of ResNet50 and VGG16 demonstrate the compatibility of LRAU with different models. Furthermore, We apply LRAU to SSD to construct a one-stage VDFR model. The proposed SSD-LRAU can detect the vehicle and recognize the fine-grained information with real-time processing speed.

To achieve real-time VReID, we proposed the CAU, which effectively extracts the discriminative local features without using any part annotation. The outstanding discriminative feature extraction ability of CAU significantly helps ResNetT-CAU achieve excellent VReID performance with very small model size. Extensive experiments on the VeRi and VehicleID datasets indicate the proposed ResNetT-CAU achieves the highest re-identification results on both datasets.

This thesis also conducts a thorough review and comparison of the state-of-the-art deep learning-based models proposed for VAVR. We present a detailed introduction to different vehicle recognition datasets used for a comprehensive evaluation of the proposed models.

We also critically discuss the major challenges and future research trends involved in each task. Our comprehensive model analysis will help researchers that are interested in VD, VMMR, and VReID, and provide them with possible directions to solve current challenges and further improve the performance and robustness of models.

6.2 Future Work

For future work, we can start with the following:

Currently, the two-stage VMMR models can recognize multiple vehicles contained in one image with a real-time processing speed on our workstation. However, the detection speed of the model may decrease on some devices with low hardware configuration. As an improvement, we can reduce the model size by quantizing or pruning [131, 264]. Then, the model can be installed on many devices, such as smartphones or vehicles with low hardware configuration.

Our one-stage VDFR method can only detect a single object once a time due to the structure of the current model. For the improvement, we can improve the model to recognize multi-object at the same time. One method is to collect a new dataset, which contains the fine-grained label for each object in one image. Then, we can directly train our model to recognize multi-object. Another method is to modify the structure of the model to make it produce multi-level labels. For example, we can design the model to recognize the general object and then recognize the fine-grained category. When the model only has a low confidence score about the fine-grained category, we can make the model only output the general category.

The current research on the realization of a reliable vision-based VReID model is still in the early stage. More efforts can be made to balance the inter-class similarity and intra-class variance to achieve a robust VReID model.

References

[1] NTOU-MMR Dataset. [Online]. Available: http://mmplab.cs.ntou.edu.tw/%20mmplab/MMR/MMR.html. Accessed on: Nov., 2018.

[2] Security Camera Warehouse. [Online]. Available: https://www.getscw.com/ip-camera/. Accessed on: May., 2020.

[3] The Industrial Liaison Program of VBIE. [Online]. Available: http://vbie.eic.nctu.edu.tw/en/introduction. Accessed on: Nov., 2018.

[4] Its canada. [Online]. Available: https://www.itscanada.ca/it/society/index.html, 2019. Accessed on: Nov., 2019.

[5] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Trans. Intell. Transp. Syst.*, 12(3):717–735, 2011.

[6] Osama Abumansoor and A. Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Trans. Veh. Technol.*, 61:275–285, 2012.

[7] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A performance evaluation of load balancing and qos-aware gateway discovery protocol for vanets. In *Proc. AINA*, pages 90–94, 2013.

[8] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A reliable quality of service aware fault tolerant gateway discovery protocol for vehicular networks. *Wirel. Commun. Mob. Comput.*, 15(10):1485–1495, 2015.

[9] Noura Aljeri, Mohammed Almulla, and Azzedine Boukerche. An efficient fault detection and diagnosis protocolfor vehicular networks. In *Proc. ACM MSWiM*, pages 23–30, 2013.

[10] Noura Aljeri and Azzedine Boukerche. Performance evaluation of movement prediction techniques for vehicular networks. In *Proc. IEEE ICC*, pages 1–6, 2017.

[11] Noura Aljeri and Azzedine Boukerche. A predictive collision detection protocol using vehicular networks. In *Proc. IEEE PIMRC*, pages 1–5. IEEE, 2017.

[12] Noura AlJeri and Azzedine Boukerche. An efficient movement-based handover prediction scheme for hierarchical mobile ipv6 in vanets. In *Proc. ACM PE-WASUN*, pages 47–54, 2018.

[13] Noura Aljeri and Azzedine Boukerche. Mobility and handoff management in connected vehicular networks. In *Proc. ACM MobiWac*, pages 82–88, 2018.

[14] Noura Aljeri and Azzedine Boukerche. A dynamic map discovery and selection scheme for predictive hierarchical mipv6 in vehicular networks. *IEEE Trans. Veh. Technol.*, 69(1):793–806, 2019.

[15] Noura Aljeri and Azzedine Boukerche. An efficient handover trigger scheme for vehicular networks using recurrent neural networks. In *Proc. ACM Q2SWinet*, pages 85–91, 2019.

[16] Noura Aljeri and Azzedine Boukerche. Movement prediction models for vehicular networks: an empirical analysis. *Wirel. Netw.*, 25(4):1505–1518, 2019.

[17] Noura Aljeri and Azzedine Boukerche. A novel online machine learning based rsu prediction scheme for intelligent vehicular networks. In *Proc. AICCSA*, pages 1–8, 2019.

[18] Noura Aljeri and Azzedine Boukerche. An optimized link duration-based mobility management scheme for connected vehicular networks. In *Proc. ACM PE-WASUN*, pages 7–14, 2019.

[19] Noura Aljeri and Azzedine Boukerche. A probabilistic neural network-based road side unit prediction scheme for autonomous driving. In *Proc. IEEE ICC*, pages 1–6, 2019.

[20] Noura Aljeri and Azzedine Boukerche. A two-tier machine learning-based handover management scheme for intelligent vehicular networks. *Ad Hoc Netw.*, 94:101930, 2019.

[21] Noura Aljeri and Azzedine Boukerche. An adaptive traffic-flow based controller deployment scheme for software-defined vehicular networks. In *Proc. ACM MSWiM*, pages 191–198, 2020.

[22] Noura Aljeri and Azzedine Boukerche. Advice-loc: An adaptive vehicle-centric location management scheme for intelligent connected cars. *Ad Hoc Netw.*, 107:102223, 2020.

[23] Noura Aljeri and Azzedine Boukerche. Fog-enabled vehicular networks: A new challenge for mobility management. *Internet Tech. Letters*, 3(6):e141, 2020.

[24] Noura Aljeri and Azzedine Boukerche. Mobility management in 5g-enabled vehicular networks: Models, protocols, and classification. *ACM Comput. Surv.*, 53(5):1–35, 2020.

[25] Noura Aljeri and Azzedine Boukerche. A performance evaluation of time-series mobility prediction for connected vehicular networks. In *Proc. ACM Q2SWinet*, pages 127–131, 2020.

[26] Thanasis Antoniou, Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris Nikoletseas, and Azzedine Boukerche. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. In *Proc. ANSS*, pages 43–52, 2004.

[27] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. ACM-SIAM SODA*, page 1027–1035, 2007.

[28] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L. Duan. Group-sensitive triplet embedding for vehicle reidentification. *IEEE Trans. Multimedia*, 20(9):2385–2399, 2018.

[29] Athanasios Bamis, Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Comput. Netw.*, 52(1):130–154, 2008.

[30] Rodolfo Bezerra Batista, Azzedine Boukerche, and Alba Cristina Magalhaes Alves de Melo. A parallel strategy for biological sequence alignment in restricted memory space. *J. Parallel Distrib. Comput.*, 68(4):548–561, 2008.

[31] M. Biglari, A. Soleimani, and H. Hassanpour. A cascaded part-based system for fine-grained vehicle classification. *IEEE Trans. Intell. Transp. Syst*, 19(1):273–283, 2018.

[32] E. Bochinski, V. Eiselein, and T. Sikora. High-speed tracking-by-detection without using image information. In *Proc. IEEE AVSS*, pages 1–6, 2017.

[33] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Comput. Commun.*, 30(11):2413–2427, 2007.

[34] A. Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks*. John Wiley & Sons, 2008.

[35] A. Boukerche, K. El-Khatib, Li Xu, and L. Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *Proc. IEEE LCN*, pages 618–624, 2004.

[36] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro. Localization systems for wireless sensor networks. *IEEE Wirel. Commun.*, 14(6):6–12, 2007.

[37] Azzedine Boukerche. A simulation based study of on-demand routing protocols for ad hoc wireless networks. In *Proc. ANSS*, pages 85–92, 2001.

[38] Azzedine Boukerche. *Handbook of algorithms for wireless networking and mobile computing*. CRC Press, 2005.

[39] Azzedine Boukerche, Noura Aljeri, Kaouther Abrougui, and Yan Wang. Towards a secure hybrid adaptive gateway discovery mechanism for intelligent transportation systems. *Secur. Commun. Netw.*, 9(17):4027–4047, 2016.

[40] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Comput. Commun.*, 29(4):477–489, 2006.

[41] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus. Energy-aware data-centric routing in microsensor networks. In *Proc. ACM MSWiM*, page 42–49, 2003.

[42] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wirel. Netw.*, 11(5):619–635, 2005.

[43] Azzedine Boukerche, Jan M Correa, Alba Cristina Magalhaes Melo, and Ricardo P Jacobi. A hardware accelerator for the fast retrieval of dialign biological sequence alignments in linear space. *IEEE Trans. Comput.*, 59(6):808–821, 2010.

[44] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Comput. Surv.*, 47(2):1–36, 2014.

[45] Azzedine Boukerche and Sajal K Das. Dynamic load balancing strategies for conservative parallel simulations. In *Proc. PADS*, pages 20–28, 1997.

[46] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks. *J. Parallel Distrib. Comput.*, 61(7):967–995, 2001.

[47] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wirel. Netw.*, 7(5):467–486, 2001.

[48] Azzedine Boukerche, Sajal K Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel pcs network simulation. In *Proc. PADS*, pages 166–173, 1999.

[49] Azzedine Boukerche, Yan Du, Jing Feng, and Richard Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *Proc. IEEE ISCC*, pages 1083–1089, 2008.

[50] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurr. Comp. Pract. E.*, 16(15):1545–1573, 2004.

[51] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks. *Comput. Commun.*, 28(10):1193–1203, 2005.

[52] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad Hoc Netw.*, 5(8):1303–1316, 2007.

[53] Azzedine Boukerche and Xin Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *Proc. IEEE GLOBECOM*, pages 5190–5194, 2007.

[54] Azzedine Boukerche, Xin Fei, and Regina B Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Comput. Commun.*, 30(14-15):2708–2720, 2007.

[55] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Netw. Appl.*, 7(2):115–126, 2002.

[56] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):911–923, 2002.

[57] Azzedine Boukerche, Kathia Regina Lemos Jucá, Joao Bosco Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Comput.*, 30(5-6):629–646, 2004.

[58] Azzedine Boukerche and Xu Li. An agent-based trust and reputation management scheme for wireless sensor networks. In *Proc. IEEE GLOBECOM*, volume 3, pages 5–pp, 2005.

[59] Azzedine Boukerche, Renato B Machado, Kathia RL Jucá, João Bosco M Sobral, and Mirela SMA Notare. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Comput. Commun.*, 30(13):2649–2660, 2007.

[60] Azzedine Boukerche, Alexander Magnano, and Noura Aljeri. Mobile ip handover for vehicular networks: Methods, models, and classifications. *ACM Comput. Surv.*, 49(4):1–34, 2017.

[61] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Netw. Appl.*, 13(6):614–626, 2008.

[62] Azzedine Boukerche, Nathan J McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Proc. ANSS*, pages 259–266, 2005.

[63] Azzedine Boukerche and Sotiris Nikoletseas. *Protocols for Data Propagation in Wireless Sensor Networks*, pages 23–51. 2004.

[64] Azzedine Boukerche and Mirela Sechi M Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *J. Parallel Distrib. Comput.*, 62(9):1476–1490, 2002.

[65] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Commun. Mag.*, 46(4):96–101, 2008.

[66] Azzedine Boukerche, Horacio A.B.F. Oliveira, Eduardo F. Nakamura, and Antonio A.F. Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Comput. Commun.*, 31(12):2838–2849, 2008.

[67] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo Freire Nakamura, and Antonio AF Loureiro. Dv-loc: a scalable localization protocol using voronoi diagrams for wireless sensor networks. *IEEE Wirel. Commun.*, 16(2):50–55, 2009.

[68] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina B Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *Proc. IEEE LCN*, pages 560–567, 2005.

[69] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proc. ACM MSWiM*, pages 157–164, 2004.

[70] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments. *J. Parallel Distrib. Comput.*, 66(4):586–599, 2006.

[71] Azzedine Boukerche, Richard WN Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Comput. Commun.*, 31(11):2716–2725, 2008.

[72] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proc. ACM PE-WASUN*, pages 88–95, 2008.

[73] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Comput. Commun.*, 31(18):4343–4351, 2008.

[74] Azzedine Boukerche and Yonglin Ren. A secure mobile healthcare system using trust-based multicast scheme. *IEEE J. Sel. Areas Commun.*, 27(4):387–399, 2009.

[75] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Proc. IEEE GLOBECOM*, pages 1–6, 2009.

[76] Azzedine Boukerche and E Robson. Vehicular cloud computing: Architectures, applications, and mobility. *Comput. Netw.*, 135:171–189, 2018.

[77] Azzedine Boukerche and Steve Rogers. Performance of gzrp ad hoc routing protocol. *J. Interconnect. Netw.*, 2(01):31–48, 2001.

[78] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *J. Parallel Distrib. Comput.*, 62(3):366–392, 2002.

[79] Azzedine Boukerche, Amber Roy, and Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *Proc. DSRT*, pages 47–54, 2000.

[80] Azzedine Boukerche and Samer Samarah. An efficient data extraction mechanism for mining association rules from wireless sensor networks. In *Proc. IEEE ICC*, pages 3936–3941, 2007.

[81] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(7):865–877, 2008.

[82] Azzedine Boukerche, Abdul Jabbar Siddiqui, and Abdelhamid Mammeri. Automated vehicle detection and classification: Models, methods, and techniques. *ACM Comput. Surv.*, 50(5):62, 2017.

[83] Azzedine Boukerche and Carl Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proc. PADS*, pages 164–172, 1994.

[84] Azzedine Boukerche and Carl Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Trans. Parallel Distrib. Syst.*, 9(8):748–757, 1998.

[85] Azzedine Boukerche and Damla Turgut. Secure time synchronization protocols for wireless sensor networks. *IEEE Wirel. Commun.*, 14(5):64–69, 2007.

[86] Azzedine Boukerche and Jiahao Wang. Machine learning-based traffic prediction models for intelligent transportation systems. *Comput. Netw.*, 181:107530, 2020.

[87] Azzedine Boukerche, Anis Zarrad, and Regina Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, 21(7):911–924, 2009.

[88] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proc. NIPS*, pages 737–744, 1993.

[89] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *Proc. ECCV*, pages 354–370, 2016.

[90] Clayson Celes, Fabrício A Silva, Azzedine Boukerche, Rossana Maria de Castro Andrade, and Antonio AF Loureiro. Improving vanet simulation with calibrated vehicular mobility traces. *IEEE Trans. Mob. Comput.*, 16(12):3376–3389, 2017.

[91] L. Chen, J. Hsieh, Y. Yan, and D. Chen. Vehicle make and model recognition using sparse representation and symmetrical surfs. In *Proc. IEEE ITSC*, pages 1143–1148, 2013.

[92] P. Chen, P. Li, Q. Li, and D. Zhang. Semi-supervised fine-grained image categorization using transfer learning with hierarchical multi-scale adversarial networks. *IEEE Access*, 7:118650–118668, 2019.

[93] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. [Online]. Available: https://arxiv.org/abs/1406.1078, 2014. Accessed on: Nov., 2018.

[94] Ruihang Chu, Yifan Sun, Yadong Li, Zheng Liu, Chi Zhang, and Yichen Wei. Vehicle re-identification with viewpoint-aware metric learning. In *Proc. IEEE ICCV*, pages 8282–8291, 2019.

[95] Sergio Correia, Azzedine Boukerche, and Rodolfo I Meneguette. An architecture for hierarchical software-defined vehicular networks. *IEEE Commun. Mag.*, 55(7):80–86, 2017.

[96] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[97] R. W. L. Coutinho, A. Boukerche, L. F. M. Vieira, and A. A. F. Loureiro. Geographic and opportunistic routing for underwater sensor networks. *IEEE Trans. Comput.*, 65(2):548–561, 2016.

[98] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Gedar: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *Proc. IEEE ICC*, pages 251–256, 2014.

[99] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. *Ad Hoc Netw.*, 34:144–156, 2015.

[100] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Design guidelines for opportunistic routing in underwater networks. *IEEE Commun. Mag.*, 54(2):40–48, 2016.

[101] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Underwater wireless sensor networks: A new challenge for topology control–based systems. *ACM Comput. Surv.*, 51(1):1–36, 2018.

[102] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Stat. Learn. in Comp. Vision*, pages 1–22, 2004.

[103] Amir Darehshoorzadeh and Azzedine Boukerche. Underwater sensor networks: A new challenge for opportunistic routing protocols. *IEEE Commun. Mag.*, 53(11):98–107, 2015.

[104] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proc. NIPS*, pages 2933–2941, 2014.

[105] Robson E De Grande and Azzedine Boukerche. Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems. *J. Parallel Distrib. Comput.*, 71(1):40–52, 2011.

[106] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Trans. Comput.*, 58(5):677–691, 2008.

[107] Z. Deng, H. Sun, S. Zhou, J. Zhao, and H. Zou. Toward fast and accurate vehicle detection in aerial images using coupled region-based convolutional neural networks. *IEEE J. Select. Top. Appl. Earth Observ. Remote Sens.*, 10(8):3652–3664, 2017.

[108] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annu. Rev. Neurosci.*, 18(1):193–222, 1995.

[109] Almudena Díaz Zayas, Delia Rico, Bruno García, and Pedro Merino. A coordination framework for experimentation in 5g testbeds: Urllc as use case. In *Proc. ACM MobiWac*, page 71–79, 2019.

[110] S. Du, M. Ibrahim, M. Shehata, and W. Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Trans. Circ. Syst. Video Technol.*, 23(2):311–325, 2013.

[111] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. [Online]. Available: https://arxiv.org/abs/1904.08189, 2019. Accessed on: Nov., 2019.

[112] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(Jul):2121–2159, 2011.

[113] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Proc. IEEE/ACM DS-RT*, pages 61–68, 2008.

[114] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In *Proc. ACM MobiWac*, pages 18–27, 2006.

[115] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proc. ACM MSWiM*, pages 165–172, 2006.

[116] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *J. Parallel Distrib. Comput.*, 68(3):321–335, 2008.

[117] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

[118] Jie Fang, Yu Zhou, Yao Yu, and Sidan Du. Fine-grained vehicle model recognition using a coarse-to-fine convolutional neural network architecture. *IEEE Trans. Intell. Transp. Syst.*, 18(7):1782–1792, 2017.

[119] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vis.*, 59(2):167–181, 2004.

[120] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proc. IEEE CVPR*, pages 4476–4484, 2017.

[121] Yang Gao, Shouyan Guo, Kaimin Huang, Jiaxin Chen, Qian Gong, Yang Zou, Tong Bai, and Gary Overett. Scale optimization for full-image-cnn vehicle detection. In *Proc. IEEE IV*, pages 785–791, 2017.

[122] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.*, 32(11):1231–1237, 2013.

[123] Baraq Ghaleb, Ahmed Y Al-Dubai, Elias Ekonomou, Ayoub Alsarhan, Youssef Nasser, Lewis M Mackenzie, and Azzedine Boukerche. A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations. *IEEE Commun. Surv. Tutor.*, 21(2):1607–1635, 2018.

[124] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE CVPR*, pages 580–587, June 2014.

[125] Ross Girshick. Fast r-cnn. In *Proc. IEEE ICCV*, pages 1440–1448, 2015.

[126] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[127] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. Imagenet auto-annotation with segmentation propagation. *Int. J. Comput. Vis.*, 110(3):328–348, 2014.

[128] H. Guo, K. Zhu, M. Tang, and J. Wang. Two-level attention network with multi-grain ranking loss for vehicle re-identification. *IEEE Trans. Image Process.*, 28(9):4328–4338, 2019.

[129] Haiyun Guo, Chaoyang Zhao, Zhiwei Liu, Jinqiao Wang, and Hanqing Lu. Learning coarse-to-fine structured feature embedding for vehicle re-identification. In *Proc. AAAI Conf. Artif. Intell.*, 2018.

[130] Hadi Habibzadeh, Tolga Soyata, Burak Kantarci, Azzedine Boukerche, and Cem Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Comput. Netw.*, 144:163–200, 2018.

[131] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. [Online]. Available: http://arxiv.org/abs/1510.00149, 2015. Accessed on: Nov., 2018.

[132] Li-Ying Hao, Jie Li, and Ge Guo. A multi-target corner pooling-based neural network for vehicle detection. *Neural Comput. Appl.*, 32(18):14497–14506, 2020.

[133] Bing He, Jia Li, Yifan Zhao, and Yonghong Tian. Part-regularized near-duplicate vehicle re-identification. In *Proc. IEEE CVPR*, pages 3997–4005, 2019.

[134] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. ECCV*, pages 770–778, 2016.

[135] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. Fastreid: A pytorch toolbox for general instance re-identification. [Online]. Available: http://arxiv.org/abs/2006.02631. Accessed on: Aug., 2020.

[136] S. He, H. Luo, W. Chen, M. Zhang, Y. Zhang, F. Wang, H. Li, and W. Jiang. Multi-domain learning and identity mining for vehicle re-identification. In *Proc. IEEE CVPRW*, pages 2485–2493, 2020.

[137] Y. He and L. Li. A novel multi-source vehicle detection algorithm based on deep learning. In *Proc. IEEE ICSIP*, pages 979–982, 2018.

[138] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. [Online]. Available: http://arxiv.org/abs/1703.07737, 2017. Accessed on: Nov., 2019.

[139] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[140] Bin Hu, Jian-Huang Lai, and Chun-Chao Guo. Location-aware fine-grained vehicle type recognition using multi-task deep networks. *Neurocomputing*, 243:60 – 68, 2017.

[141] Qichang Hu, Huibing Wang, Teng Li, and Chunhua Shen. Deep cnns with spatially weighted pooling for fine-grained car recognition. *IEEE Trans. Intell. Transp. Syst.*, 18(11):3147–3156, 2017.

[142] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proc. IEEE CVPR*, pages 2261–2269, 2017.

[143] Kun Huang and Bailing Zhang. Fine-grained vehicle recognition by deep convolutional neural network. In *Proc. CISP-BMEI*, pages 465–470, 2016.

[144] IEEE Standards Association. Ieee 802.11p-2010 - ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments. [Online]. Available: https://standards.ieee.org/standard/802_11p-2010.html. Accessed on: Nov, 2019.

[145] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. [Online]. Available: https://arxiv.org/abs/1502.03167, 2015. Accessed on: Nov., 2018.

[146] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. [Online]. Available: http://arxiv.org/abs/1502.03167, 2015. Accessed on: Nov., 2017.

[147] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. [Online]. Available: http://arxiv.org/abs/1506.02025, 2015. Accessed on: Nov., 2019.

[148] Saumya Jetley, Nicholas A. Lord, Namhoon Lee, and Philip H. S. Torr. Learn to pay attention. [Online]. Available: http://arxiv.org/abs/1804.02391, 2018. Accessed on: Sep., 2018.

[149] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM Int. Conf. Multimed.*, pages 675–678, 2014.

[150] N. Jiang, Y. Xu, Z. Zhou, and W. Wu. Multi-attribute driven vehicle re-identification with spatial-temporal re-ranking. In *Proc. IEEE ICIP*, pages 858–862, 2018.

[151] Q. Jiang, L. Cao, M. Cheng, C. Wang, and J. Li. Deep neural networks-based vehicle detection in satellite images. In *Proc. ISBB*, pages 184–187, 2015.

[152] Sultan Daud Khan and Habib Ullah. A survey of advances in vision-based vehicle re-identification. *Comput. Vis. Image Underst.*, 182:50 – 63, 2019.

[153] P. Khorramshahi, A. Kumar, N. Peri, S. S. Rambhatla, J. Chen, and R. Chellappa. A dual-path model with adaptive attention for vehicle re-identification. In *Proc. IEEE ICCV*, pages 6131–6140, 2019.

[154] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. [Online]. Available: https://arxiv.org/abs/1412.6980, 2014. Accessed on: Nov., 2018.

[155] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *Proc. IEEE CVPR*, pages 7025–7034, 2017.

[156] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Proc. IEEE CVPR*, pages 5546–5555, 2015.

[157] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proc. IEEE CVPR*, pages 554–561, 2013.

[158] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

[159] R. Kuma, E. Weill, F. Aghdasi, and P. Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. In *Proc. IEEE IJCNN*, pages 1–9, 2019.

[160] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proc. ECCV*, pages 734–750, 2018.

[161] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE Proc. IRE* (through 1962)*, 86(11):2278–2324, 1998.

[162] Suichan Li and Feng Chen. 3D-DETNet: a single stage video-based vehicle detector. In *Proc. IWPR*, volume 10828, pages 60 – 66, 2018.

[163] X. Li, L. Yu, D. Chang, Z. Ma, and J. Cao. Dual cross-entropy loss for small-sample fine-grained vehicle classification. *IEEE Trans. Veh. Technol.*, 68(5):4204–4212, 2019.

[164] Y. Li, F. Naeimipoor, and A. Boukerche. Video dissemination protocols in urban vehicular ad hoc network: A performance evaluation study. In *Proc. IEEE WCNC*, pages 2611–2616, 2014.

[165] Jin-Fu Lin, Yen-Liang Lin, Erh-Kan King, Hung-Ting Su, and Winston H Hsu. Cross-domain hallucination network for fine-grained object recognition. In *Proc. IEEE CVPRW*, pages 1214–1221, 2018.

[166] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear convolutional neural networks for fine-grained visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(6):1309–1322, 2018.

[167] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, pages 740–755, 2014.

[168] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proc. IEEE ICCV*, pages 1449–1457, 2015.

[169] C. Liu, H. Xie, Z. Zha, L. Yu, Z. Chen, and Y. Zhang. Bidirectional attention-recognition model for fine-grained object classification. *IEEE Trans. Multimedia*, 22(7):1785–1795, 2020.

[170] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proc. IEEE CVPR*, pages 2167–2175, 2016.

[171] K. Liu and G. Mattyus. Fast multiclass vehicle detection on aerial images. *IEEE Geosci. Remote Sens. Lett.*, 12(9):1938–1942, 2015.

[172] Songtao Liu, Di Huang, and Yunhong Wang. Receptive field block net for accurate and fast object detection. [Online]. Available: http://arxiv.org/abs/1711.07767. Accessed on: Sep., 2018.

[173] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. ECCV*, pages 21–37, 2016.

[174] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[175] X. Liu, W. Liu, H. Ma, and H. Fu. Large-scale vehicle re-identification in urban surveillance videos. In *Proc. IEEE ICME*, pages 1–6, 2016.

[176] X. Liu, W. Liu, T. Mei, and H. Ma. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Trans. Multimedia*, 20(3):645–658, 2018.

[177] X. Liu, S. Zhang, Q. Huang, and W. Gao. Ram: A region-aware deep model for vehicle re-identification. In *Proc. IEEE ICME*, pages 1–6, 2018.

[178] Xiao Liu, Jiang Wang, Shilei Wen, Errui Ding, and Yuanqing Lin. Localizing by describing: Attribute-guided attention localization for fine-grained recognition. In *Proc. AAAI Conf. Artif. Intell.*, number 1, 2017.

[179] Xiao Liu, Tian Xia, Jiang Wang, Yi Yang, Feng Zhou, and Yuanqing Lin. Fully convolutional attention networks for fine-grained recognition. [Online]. Available: https://arxiv.org/abs/1603.06765, 2016. Accessed on: Sep., 2018.

[180] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *Proc. ECCV*, pages 869–884, 2016.

[181] Dingkun Long, Richong Zhang, and Yongyi Mao. Prototypical recurrent unit. *Neurocomputing*, 311:146–154, 2018.

[182] Y. Lou, Y. Bai, J. Liu, S. Wang, and L. Duan. Embedding adversarial learning for vehicle re-identification. *IEEE Trans. Image Process.*, 28(8):3794–3807, 2019.

[183] Yihang Lou, Yan Bai, Jun Liu, Shiqi Wang, and Lingyu Duan. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In *Proc. IEEE CVPR*, pages 3235–3243, 2019.

[184] L. Lu and H. Huang. A hierarchical scheme for vehicle make and model recognition from frontal images of vehicles. *IEEE Trans. Intell. Transp. Syst*, 20(5):1774–1786, 2019.

[185] X. Ma and A. Boukerche. An ai-based visual attention model for vehicle make and model recognition. In *Proc. IEEE ISCC*, pages 1–6, 2020.

[186] Z. Ma, D. Chang, J. Xie, Y. Ding, S. Wen, X. Li, Z. Si, and J. Guo. Fine-grained vehicle classification with channel max pooling modified cnns. *IEEE Trans. Veh. Technol.*, 68(4):3224–3233, 2019.

[187] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[188] A. Mammeri, E. Khiari, and A. Boukerche. Road-sign text recognition architecture for intelligent transportation systems. In *Proc. IEEE VTC*, pages 1–5, 2014.

[189] A. Mammeri, G. Lu, and A. Boukerche. Design of lane keeping assist system for autonomous vehicles. In *Proc. NTMS*, pages 1–5, 2015.

[190] A. Mammeri, D. Zhou, and A. Boukerche. Animal-vehicle collision mitigation system for automated vehicles. *IEEE Trans. Syst., Man, Cybern. Syst*, 46(9):1287–1299, 2016.

[191] A. Mammeri, D. Zhou, A. Boukerche, and M. Almulla. An efficient animal detection system for smart cars using cascaded classifiers. In *Proc. IEEE ICC*, pages 1854–1859, 2014.

[192] A. Mammeri, T. Zuo, and A. Boukerche. Extending the detection range of vision-based driver assistance systems application to pedestrian protection system. In *Proc. IEEE GLOBECOM*, pages 1358–1363, 2014.

[193] A. Mammeri, T. Zuo, and A. Boukerche. Extending the detection range of vision-based vehicular instrumentation. *IEEE Trans. Instrum. Meas.*, 65(4):856–873, 2016.

[194] Abdelhamid Mammeri, Azzedine Boukerche, and Guangqian Lu. Lane detection and tracking system based on the mser algorithm, hough transform and kalman filter. In *Proc. ACM MSWiM*, page 259–266, 2014.

[195] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. A real-time lane marking localization, tracking and communication system. *Comput. Commun.*, 73:132–143, 2016.

[196] Anahit Martirosyan, Azzedine Boukerche, and Richard Pazzi. A taxonomy of cluster-based routing protocols for wireless sensor networks. In *Proc. ISPAN*, pages 247–253, 2008.

[197] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. [Online]. Available: http://arxiv.org/abs/1611.06440. Accessed on: May, 2020.

[198] Ahmed Mostefaoui, Mahmoud Melkemi, and Azzedine Boukerche. Localized routing approach to bypass holes in wireless sensor networks. *IEEE Trans. Comput.*, 63(12):3053–3065, 2013.

[199] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Proc. NeurIPS*, pages 4694–4703, 2019.

[200] F. Naeimipoor and A. Boukerche. A hybrid video dissemination protocol for vanets. In *Proc. IEEE ICC*, pages 112–117, 2014.

[201] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[202] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proc. ECCV*, pages 483–499, 2016.

[203] Horacio ABF Oliveira, Azzedine Boukerche, Eduardo F Nakamura, and Antonio AF Loureiro. Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Perform. Eval.*, 66(3-5):209–222, 2009.

[204] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proc. ACM GIS*, pages 71–78, 2005.

[205] Renê Oliveira, Carlos Montez, Azzedine Boukerche, and Michelle S Wangham. Reliable data dissemination protocol for vanet traffic safety applications. *Ad Hoc Netw.*, 63:30–44, 2017.

[206] Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. Automatic differentiation in pytorch. In *Proc. NIPS*, pages 1–4, 2017.

[207] Richard W Pazzi and Azzedine Boukerche. Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices. *ACM Trans. Multimed. Comput. Commun. Appl.*, 11(1):1–22, 2014.

[208] Richard WN Pazzi and Azzedine Boukerche. Mobile data collector strategy for delay-sensitive applications over wireless sensor networks. *Comput. Commun.*, 31(5):1028–1039, 2008.

[209] G. Pearce and N. Pears. Automatic make and model recognition from frontal images of cars. In *Proc. IEEE AVSS*, pages 373–378, 2011.

[210] Jingjing Qian, Wei Jiang, Hao Luo, and Hongyan Yu. Stripe-based and attribute-aware network: a two-branch deep model for vehicle re-identification. *Meas. Sci. Technol.*, 31(9):095401, 2020.

[211] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1):145–151, 1999.

[212] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. [Online]. Available: http://arxiv.org/abs/1511.06434, 2015. Accessed on: Nov., 2019.

[213] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE CVPR*, pages 779–788, 2016.

[214] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proc. IEEE CVPR*, pages 7263–7271, 2017.

[215] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. [Online]. Available: https://arxiv.org/abs/1804.02767, 2018. Accessed on: Nov., 2018.

[216] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. NIPS*, pages 91–99, 2015.

[217] Y. Ren, R. Werner, N. Pazzi, and A. Boukerche. Monitoring patients via a secure and mobile healthcare system. *IEEE Wirel. Commun.*, 17(1):59–65, 2010.

[218] Yonglin Ren and Azzedine Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Proc. IEEE ICC*, pages 2129–2133, 2008.

[219] Cristiano Rezende, Azzedine Boukerche, Heitor S Ramos, and Antonio AF Loureiro. A reactive and scalable unicast solution for video streaming over vanets. *IEEE Trans. Comput.*, 64(3):614–626, 2014.

[220] Cristiano Rezende, Abdelhamid Mammeri, Azzedine Boukerche, and Antonio AF Loureiro. A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection. *Ad Hoc Netw.*, 17:1–17, 2014.

[221] E. Ristani and C. Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proc. IEEE CVPR*, pages 6036–6046, 2018.

[222] Pau Rodríguez, Josep M Gonfaus, Guillem Cucurull, F XavierRoca, and Jordi Gonzalez. Attend and rectify: a gated attention mechanism for fine-grained recovery. In *Proc. ECCV*, pages 349–364, 2018.

[223] Sebastian Ruder. An overview of gradient descent optimization algorithms. [Online]. Available: https://arxiv.org/abs/1609.04747, 2016. Accessed on: Nov., 2018.

[224] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[225] SAE International. Dedicated short range communications (dsrc) message set dictionary J2735. [Online]. Available: https://www.sae.org/standards/content/j2735_201603/. Accessed on: Nov, 2019.

[226] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Trans. Veh. Technol.*, 60(2):656–663, 2010.

[227] Jun Sang, Zhongyuan Wu, Pei Guo, Haibo Hu, Hong Xiang, Qian Zhang, and Bin Cai. An improved yolov2 for vehicle detection. *Sensors*, 18(12):4272, 2018.

[228] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang. Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. In *Proc. IEEE ICCV*, pages 1918–1927, 2017.

[229] A. J. Siddiqui, A. Mammeri, and A. Boukerche. Real-time vehicle make and model recognition based on a bag of surf features. *IEEE Trans. Intell. Transp. Syst.*, 17(11):3205–3219, 2016.

[230] A. J. Siddiqui, A. Mammeri, and A. Boukerche. Real-time vehicle make and model recognition based on a bag of surf features. *IEEE Trans. Intell. Transp. Syst*, 17(11):3205–3219, 2016.

[231] Fabricio A Silva, Azzedine Boukerche, Thais RM Braga Silva, Linnyer B Ruiz, Eduardo Cerqueira, and Antonio AF Loureiro. Vehicular networks: A new challenge for content-delivery-based applications. *ACM Comput. Surv.*, 49(1):1–29, 2016.

[232] A. Simonelli, F. De Natale, S. Messelodi, and S. R. Bulo. Increasingly specialized ensemble of convolutional neural networks for fine-grained recognition. In *Proc. IEEE ICIP*, pages 594–598, 2018.

[233] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. [Online]. Available: https://arxiv.org/abs/1409.1556, 2014. Accessed on: Nov., 2017.

[234] S. Sivaraman and M. M. Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Trans. Intell. Transp. Syst.*, 14(4):1773–1795, 2013.

[235] Richard Socher, Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE CVPR*, pages 248–255, 2009.

[236] J. Sochor, A. Herout, and J. Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *Proc. IEEE CVPR*, pages 3006–3015, 2016.

[237] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *Proc. IEEE CVPR*, pages 5686–5696, 2019.

[238] P. Sun and A. Boukerche. Challenges of designing computer vision-based pedestrian detector for supporting autonomous driving. In *Proc. IEEE MASS*, pages 28–36, 2019.

[239] Peng Sun, Noura AlJeri, and Azzedine Boukerche. A novel passive road side unit detection scheme in vehicular networks. In *Proc. IEEE GLOBECOM*, pages 1–5, 2017.

[240] Peng Sun, Noura AlJeri, and Azzedine Boukerche. An energy-efficient proactive handover scheme for vehicular networks based on passive rsu detection. *IEEE Trans. Sustain. Comput.*, 5(1):37–47, 2018.

[241] Peng Sun, Noura AlJeri, and Azzedine Boukerche. Dacon: A novel traffic prediction and data-highway-assisted content delivery protocol for intelligent vehicular networks. *IEEE Trans. Sustain. Comput.*, 5(4):501–513, 2020.

[242] Peng Sun, Noura Aljeri, and Azzedine Boukerche. Machine learning-based models for real-time traffic flow prediction in vehicular networks. *IEEE Netw.*, 34(3):178–185, 2020.

[243] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE CVPR*, pages 2818–2826, 2016.

[244] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. ECCV*, pages 1–9, 2015.

[245] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Proc. ICANN*, pages 270–279, 2018.

[246] M. Tan, G. Wang, J. Zhou, Z. Peng, and M. Zheng. Fine-grained classification via hierarchical bilinear pooling with aggregated slack mask. *IEEE Access*, 7:117944–117953, 2019.

[247] Zheng Tang, Milind Naphade, Stan Birchfield, Jonathan Tremblay, William Hodge, Ratnesh Kumar, Shuo Wang, and Xiaodong Yang. Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data. In *Proc. IEEE ICCV*, pages 211–220, 2019.

[248] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David C. Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. [Online]. Available: http://arxiv.org/abs/1903.09254, 2019. Accessed on: Nov., 2019.

[249] Yanling Tian, Weitong Zhang, Qieshi Zhang, Gang Lu, and Xiaojun Wu. Selective multi-convolutional region feature extraction based iterative discrimination cnn for fine-grained vehicle model recognition. In *Proc. IEEE ICPR*, pages 3279–3284, 2018.

[250] Thang To, Jonathan Tremblay, Duncan McKay, Yukie Yamaguchi, Kirby Leung, Adrian Balanon, Jia Cheng, William Hodge, and Stan Birchfield. NDDS: NVIDIA deep learning dataset synthesizer. [Online]. Available: https://github.com/NVIDIA/Dataset_Synthesizer, 2018. Accessed on: Nov., 2019.

[251] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *Int. J. Comput. Vis.*, 104(2):154–171, 2013.

[252] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *Proc. IEEE CVPR*, pages 6450–6458, 2017.

[253] R. Wang, M. Almulla, C. Rezende, and A. Boukerche. Video streaming over vehicular networks by a multiple path solution with error correction. In *Proc. IEEE ICC*, pages 580–585, 2014.

[254] Renfei Wang, Cristiano Rezende, Heitor S Ramos, Richard W Pazzi, Azzedine Boukerche, and Antonio AF Loureiro. Liaithon: A location-aware multipath video streaming scheme for urban vehicular networks. In *Proc. IEEE ISCC*, pages 436–441, 2012.

[255] Y. Wang, V. I. Morariu, and L. S. Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *Proc. IEEE CVPR*, pages 4148–4157, 2018.

[256] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *Proc. IEEE ICCV*, pages 379–387, 2017.

[257] C. Wu, C. Liu, C. Chiang, W. Tu, and S. Chien. Vehicle re-identification with the space-time prior. In *Proc. IEEE CVPRW*, pages 121–1217, 2018.

[258] F. Wu, S. Yan, J. S. Smith, and B. Zhang. Joint semi-supervised learning and re-ranking for vehicle re-identification. In *Proc. IEEE ICPR*, pages 278–283, 2018.

[259] L. Wu, Y. Wang, X. Li, and J. Gao. Deep attention-based spatially recursive networks for fine-grained visual recognition. *IEEE Trans. Biomed. Circ. Syst.*, 49(5):1791–1802, 2019.

[260] Zhongyuan Wu, Jun Sang, Qian Zhang, Hong Xiang, Bin Cai, and Xiaofeng Xia. Multi-scale vehicle detection for foreground-background class imbalance with improved yolov2. *Sensors*, 19(15):3336, 2019.

[261] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proc. IEEE CVPR*, pages 842–850, 2015.

[262] H. Xie, A. Boukerche, and A. A. F. Loureiro. A multipath video streaming solution for vehicular networks with link disjoint and node-disjoint. *IEEE Trans. Parallel Distrib. Syst.*, 26(12):3223–3235, 2015.

[263] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Proc. IEEE CVPR*, pages 2645–2654, 2015.

[264] Jiaolong Xu, Yiming Nie, Peng Wang, and Antonio M López. Training a binary weight object detector by knowledge transfer for autonomous driving. In *Proc. IEEE ICRA*, pages 2379–2384, 2019.

[265] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proc. IEEE CVPR*, pages 3973–3981, 2015.

[266] M. B. Younes and A. Boukerche. A vehicular network based intelligent lane change assistance protocol for highways. In *Proc. IEEE ICC*, pages 1–6, 2017.

[267] Maram Bani Younes and Azzedine Boukerche. Intelligent traffic light controlling algorithms using vehicular networks. *IEEE Trans. Veh. Technol.*, 65(8):5887–5899, 2015.

[268] Maram Bani Younes and Azzedine Boukerche. A performance evaluation of an efficient traffic congestion detection protocol (ecode) for intelligent transportation systems. *Ad Hoc Netw.*, 24:317–336, 2015.

[269] Maram Bani Younes and Azzedine Boukerche. An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wirel. Netw.*, 24(7):2451–2463, 2018.

[270] Chaojian Yu, Xinyi Zhao, Qi Zheng, Peng Zhang, and Xinge You. Hierarchical bilinear pooling for fine-grained visual recognition. In *Proc. ECCV*, pages 595–610, 2018.

[271] Ye Yu, Qiang Jin, and Chang Wen Chen. Ff-cmnet: A cnn-based model for fine-grained classification of car models based on feature fusion. In *Proc. IEEE ICME*, pages 1–6, 2018.

[272] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. [Online]. Available: http://arxiv.org/abs/1605.07146. Accessed on: Nov., 2018.

[273] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. [Online]. Available: http://arxiv.org/abs/1612.03928, 2016. Accessed on: Nov., 2018.

[274] Nagina Zarin and Anjali Agarwal. Qos based joint radio resource allocation for multihoming calls in heterogeneous wireless access network. In *Proc. ACM MobiWac*, page 37–42, 2018.

[275] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Proc. IEEE CVPR*, pages 2528–2535, 2010.

[276] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, pages 818–833, 2014.

[277] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *Proc. IEEE CVPR*, pages 1143–1152, 2016.

[278] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *Proc. ECCV*, pages 834–849, 2014.

[279] Qiang Zhang, Li Zhuo, Shiyu Zhang, Jiafeng Li, Hui Zhang, and Xiaoguang Li. Fine-grained vehicle recognition using lightweight convolutional neural network with combined learning strategy. In *Proc. IEEE BigMM*, pages 1–5, 2018.

[280] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *Proc. IEEE CVPR*, pages 1114–1123, 2016.

[281] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *Proc. IEEE CVPR*, pages 1134–1142, 2016.

[282] Y. Zhang, D. Liu, and Z. Zha. Improving triplet-wise training of convolutional neural network for vehicle re-identification. In *Proc. IEEE ICME*, pages 1386–1391, 2017.

[283] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Comput. Netw.*, 54(4):558–572, 2010.

[284] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Trans. Multimedia*, 19(6):1245–1256, 2017.

[285] Dongbin Zhao, Yaran Chen, and Le Lv. Deep reinforcement learning with visual attention for vehicle classification. *IEEE Trans. Cogn. Develop. Syst.*, 9(4):356–367, 2017.

[286] H. Zheng, J. Fu, Z. Zha, J. Luo, and T. Mei. Learning rich part hierarchies with progressive attention networks for fine-grained image recognition. *IEEE Trans. Image Process.*, 29:476–488, 2020.

[287] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proc. IEEE ICCV*, volume 6, 2017.

[288] Z. Zheng, L. Zheng, and Y. Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proc. IEEE ICCV*, pages 3774–3782, 2017.

[289] Dunhao Zhong, Peng Sun, and Azzedine Boukerche. Empirical study and analysis of the impact of traffic flow control at road intersections on vehicle energy consumption. In *Proc. ACM MobiWac*, page 21–28, 2020.

[290] Weilin Zhong, Linfeng Jiang, Tao Zhang, Jinsheng Ji, and Huilin Xiong. A multi-part convolutional attention network for fine-grained image recognition. In *Proc. IEEE ICPR*, pages 1857–1862, 2018.

[291] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proc. AAAI Conf. Artif. Intell.*, pages 13001–13008, 2020.

[292] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. IEEE CVPR*, pages 2921–2929, 2016.

[293] Y. Zhou, L. Liu, and L. Shao. Vehicle re-identification by deep hidden multi-view inference. *IEEE Trans. Image Process.*, 27(7):3275–3287, 2018.

[294] Y. Zhouy and L. Shao. Viewpoint-aware attentive multi-view inference for vehicle re-identification. In *Proc. IEEE CVPR*, pages 6489–6498, 2018.

[295] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. IEEE ICCV*, pages 2242–2251, 2017.

[296] J. Zhu, H. Zeng, J. Huang, S. Liao, Z. Lei, C. Cai, and L. Zheng. Vehicle re-identification using quadruple directional deep learning features. *IEEE Trans. Intell. Transp. Syst.*, 21(1):410–420, 2020.

[297] J. Zhu, H. Zeng, Z. Lei, S. Liao, L. Zheng, and C. Cai. A shortly and densely connected convolutional neural network for vehicle re-identification. In *Proc. IEEE ICPR*, pages 3285–3290, 2018.

[298] Jianqing Zhu, Yongzhao Du, Yang Hu, Lixin Zheng, and Canhui Cai. Vrsdnet: vehicle re-identification with a shortly and densely connected convolutional neural network. *Multimedia Tools Appl.*, 78(20):29043–29057, 2019.