DVHN: A Deep Hashing Framework for Large-Scale Vehicle Re-Identification

Yongbiao Chen®, Sheng Zhang, Fangxin Liu®, Chenggang Wu, Kaicheng Guo, and Zhengwei Qi®

Abstract—Vehicle re-identification is a pervasive technology in real-world intelligence transportation systems. Conventional methods generally perform re-identification tasks by representing vehicle images as real-valued feature vectors and then ranking the gallery images by computing the corresponding Euclidean distances. Despite achieving remarkable retrieval accuracy, these high-dimensional real-valued feature vectors are not tailored for fast indexing and matching and require tremendous memory and computation when the gallery set is large, making them inapplicable in a large-scale real-world retrieval setting. In light of this limitation, in this paper, we make the very first attempt to develop an efficient vehicle re-identification system (DVHN) for real-world large-scale retrieval tasks with deep hashing learning. It could substantially reduce memory usage and enhances retrieval efficiency while maintaining retrieval accuracy. Concretely, DVHN directly learns discrete compact binary hashing codes for each image by jointly optimizing the feature learning network and the hash code generating module. Specifically, we directly constrain the output from the convolutional neural network to be discrete binary codes and ensure the learned binary codes are optimal for classification. To optimize the deep discrete hashing framework, we further propose an alternating minimization method for learning binary similarity-preserved hashing codes. Extensive experiments on two widely-studied vehicle re-identification datasets—VehicleID and VeRi—have demonstrated the superiority of our method against the state-of-the-art deep hash methods. DVHN of 2048 bits can achieve 13.94% and 10.21% accuracy improvement in terms of mAP and Rank@1 for VehicleID (800) dataset. For VeRi, we achieve 35.45% and 32.72% performance gains for Rank@1 and mAP, respectively.

Index Terms—Deep hashing, vehicle re-identification, approximate nearest neighbor search, deep learning.

I. INTRODUCTION

Vehicle re-identification (vehicle ReID) [1], [2], [3], [4], [5], [6] has been receiving growing attention among the computer vision research community for its pervasive uses in Intelligent Transportation Systems. It targets retrieving the images of a vehicle of interest in a large image database captured by non-overlapping surveillance cameras. The general re-identification framework consists of two modules: feature learning and metric learning. 1) the feature learning module is responsible for extracting discriminative features from vehicle images. 2) the distance metric learning module [7] is for preserving the distances of original images in the embedding space. Previous vehicle re-identification methods have achieved pronounced performances on widely-studied research datasets. Nonetheless, these traditional vehicle re-identification methods are not applicable in a real-world setting where the gallery normally contains an astronomical amount of vehicle images. The main weaknesses of these traditional vehicle re-identification methods are as follows. First, the storage cost of high-dimensional feature vectors could be exceedingly high when there exists an extremely large number of gallery vehicle images. For instance, storing a 2048-dimensional feature vector of data type ‘float64’ takes up 16 kilobytes. For a gallery set of 10 million vehicle images, the total memory storage cost could be up to 150 gigabytes. On top of that, the Euclidean distance computation between two high-dimensional feature vectors could be time-consuming [8], making it undesirable in a scenario where the retrieval speed is of critical importance.

Recently, deep hashing methods have been widely adopted for large-scale image retrieval due to their low storage cost and high retrieval efficiency. The goal of deep hash is to learn a hashing function that embeds images into compact binary hash codes in the hamming space while preserving their similarity in the original space [8], [9], [10], [11]. Since a 2048-bit hamming code only takes up 256 bytes in memory, the total storage for a dataset of 10 million images is less than 2.4 gigabytes, saving up to 147 gigabytes of memory compared to using real-valued feature vectors. Further, as stated in [12], the computation of Hamming distance between binary hash codes can be accelerated by using the built-in CPU hardware instruction—XOR. Generally, the hamming distance computation could be completed with several machine instructions, significantly faster than computing its euclidean distance counterpart.

Captive by the before-mentioned benefits, one may naturally ponder the possibility of directly applying the off-the-shelf deep hash techniques into addressing the large-scale vehicle ReID problem. Although researchers have successfully applied deep hashing in image retrieval, the unique features of the ReID task make it non-trivial to apply these methods directly, usually with notable performance drops. The degraded performances could be ascribed to the fact that
general-purposed deep hashing methods fail to learn robust and discriminative features of vehicle ReID datasets. For instance, the canonical deep hashing method - SFH [13] is the first end-to-end supervised deep hashing method that employs a triplet-ranking on the hashing codes produced by the divide-and-encode module. DSH employs a hinge loss on image pairs to enforce the outputs for two similar images to stay close while dissimilar pairs to stay far away. Meanwhile, they adopt L1-norm on the outputs from the convolutional neural network to produce binary hashing codes. However, these methods are tailored for retrieval settings where the image database contains images from a very limited number of classes. In the vehicle re-identification scenario where there are thousands, even millions of vehicle ids, and where viewpoint variation for each vehicle identity is considerable, these off-the-shelf deep hash methods can only learn sub-optimal feature representation, leading to sub-optimal hash codes.

In this paper, we introduce a novel deep hashing-based framework for efficient large-scale vehicle ReID, dubbed DVHN, which fully exploits the merits of deep hashing learning while preserving the retrieval accuracy in large-scale vehicle re-identification tasks. Specifically, it consists of four modules: ① unlike traditional hashing methods which rely on easy image pairs or triplets, we propose an on-the-fly hard-negative mining random triplet generation module, ② a convolutional neural network-based feature extraction module, ③ a hash code generation module that adopts metric learning to enforce the learned hash codes to preserve the visual similarity and dissimilarity in the original raw pixel space and ④ a discrete Hamming hashing learning module. Regarding the similarity-preserving metric learning module, concretely, we adopt a batch-hard triplet loss [14] to pull close visually-similar image pairs and push away visually-disparate images in the hamming space. In addition, to further propel the learned hash codes to capture more discriminative features, we adopt Identity loss which is essentially a cross-entropy classification loss by regarding each vehicle identity as a different class. As for the discrete Hamming hashing module, we constrain the continuous feature into binary codes directly with a sign activation function. Subsequently, we train another classifier on the binary codes to ensure the learned binary hamming codes are optimal for classification. Since the sign function is no-differentiable, we propose to adopt a novel optimization method to train the DVHN framework, dubbed alternative optimization. In the test stage, for all the gallery and query images, we encode them into binary Hamming codes and rank the gallery images according to their corresponding hamming distances from the query image.

To sum up, we make the following contributions:

1) We analyze the key obstacles impeding applying existing vehicle ReID methods into the real-world, large-scale vehicle re-identification setting. We make the very first attempt to conduct efficient fast vehicle re-identification with deep hashing learning.

2) We propose a novel framework that learns the compact hamming code for vehicle images in an end-to-end manner. We propose a novel discrete Hamming hashing learning module, which learns the binary Hamming codes while ensuring the binary codes are ideal for classification. To tackle the challenging discrete optimization problem, we propose to adopt a novel alternative optimization scheme to train the framework.

3) We implement several state-of-the-art deep hashing methods with Pytorch, download the open-sourced methods, and thoroughly test their performances on the standard vehicle ReID datasets. We trained 72 models from scratch and provided valuable data for research in this literature. Through comprehensive comparisons, we have demonstrated the effectiveness of our method and evidenced our superiority against the existing state-of-the-art methods.

II. RELATED WORKS

In this section, we elaborate on the previous research works that are related to our work. Specifically, we divide them into two categories according to whether they adopt binary feature vectors or not.

A. Hash for Content Retrieval

Recent years have seen a surge in attention around the theme of hashing for large-scale content retrieval [15], [16], [17], [18], [19], [20], [21], [22]. According to the way they extract the features, existing hashing methods could be categorized into two groups: shallow hashing methods and deep learning-based-hash methods.

Typical shallow methods are reliant upon the handcrafted features to learn hash codes. A canonical example is [15], which seeks to find a locality-sensitive hash family where the probability of hash collisions for similar objects is much higher than those for dissimilar ones. Later, based on LSH, [23] further developed another variant of LSH (dubbed SIMHASH) for cosine similarities in Euclidean space. Though effective, LSH-based hash methods require long code lengths to generate satisfactory performance. In 2008, [16], proposes to learn compact hash codes by minimizing the weighted Hamming distance of image pairs. Though these handcrafted feature-based shallow methods achieved success to some extent, when applied to real data where dramatic appearance variation exists, they generally fail to capture the discriminative semantic information, leading to compromised performances.

In light of this dilemma, a wealth of deep learning-based hash methods has been proposed [11], [24], [25], [26], [27]. According to whether supervisory signals are used, deep hashing methods could be classified into two categories: the supervised deep hashing methods and unsupervised deep hashing methods. Unsupervised deep hashing methods project images into binary hashing codes by exploring the intrinsic Data characteristics. Canonical examples are [28], [29], [30], [31], and [32]. Specifically, [28] introduces a binary generative adversarial network to embed images into binary codes in an unsupervised way. Reference [29] proposes to learn a distilled dataset consisting of image pairs, which serve as the similarity signals. Reference [30] introduces a scheme by exploring the intrinsic manifold structure of the features and reconstructing the local semantic similarity structure with
the cosine similarity of data points. Reference [31] proposes to adopt contrastive learning with information bottleneck to conduct unsupervised hashing. Reference [32] further proposes to obtain reliable and smooth guidance through global refinement and similarity statistical distribution. Our work falls into the supervised hashing category. Concretely, the supervisory signals are used to construct a similarity matrix. In 2014, [33] introduces a two-stage hashing scheme that first decomposes the similarity matrix \( S \) into a product of \( H \) and \( H^T \) where each row of it is the hash code for a specific training image. Then, in the later stage, the convolutional neural network is trained with the supervision signals from the learned hash codes. Later, [13] proposes to learn the features and hash codes in an end-to-end manner. Reference [26] proposes a Bayesian learning framework adopting pairwise loss for similarity preserving. Reference [34] further proposes to substitute the previous probability generation function for neural network output logits with a Cauchy distribution to penalize similar image pairs with hamming distances larger than a threshold. Reference [35] introduced a deep incremental hash framework that alleviates the burden of retraining the model when the gallery image set increases. Reference [24] further introduces a deep polarized loss for the hamming code generation, obviating the need for an additional quantization loss. Despite the successes, these methods are not tailored for the large-scale vehicle re-identification scenario and could only achieve sub-optimal performances, as illustrated before.

B. Vehicle ReID

A wealth of vehicle ReID methods [6], [27], [36], [37], [38] [39], [40], [41], [42] have been introduced in recent years given their extensive uses in digital surveillance, intelligent transportation [43], urban computing [44]. It is challenging considering its large intra-class variation and small inter-class differences. Some works strive to boost the discriminability of the learned representations. Reference [1] proposes a two-stream network to project vehicle images into comparable euclidean space. Reference [3] proposes to learn feature representation by exploiting multi-grain ranking constraints. Reference [4] further proposes to learn robust features in a coarse-to-fine fashion. In 2019, [45] adopts vehicle part information to assist in global feature learning. Reference [5] introduces a novel end-to-end adversarial learning-based network to learn robust and discriminative features. [46] proposes to construct another large dataset named “VehicleNet” and innovated a two-stage learning framework to learn robust feature representation. Reference [47] proposes to learn robust features with a part-aware structure-based vehicle re-identification system. Reference [48] introduces a super-resolution-based part collaboration network (SPCN) for vehicle ReID, which focuses on local features with strong differentiation.

Another line of research focuses on learning viewpoint-invariant embedding for re-identification. Canonical examples include [37], [39], [49], and [50]. Reference [37] proposes to mine the keypoint localization information to align and extract viewpoint-invariant information. Reference [49] further introduces a scheme that utilizes single-view input information to generate multi-view features. Reference [50] innovates a pose-aware multi-task learning approach to learn viewpoint-invariant features. Reference [39] introduces to disentangle the feature representation into orientation-specific and orientation-invariant features and, in the later stage, uses the orientation-invariant feature to perform the vehicle ReID task. Reference [51] proposes to extract quadruple directional deep features for vehicle-re identification while [52] proposes an end-to-end distance-based deep neural network combining multi-regional features to learn to distinguish local and global differences of a vehicle image. Reference [53] explores vehicle parsing to learn discriminative part-level features and build a part-neighboring graph to model the correlation between every part feature. To alleviate the influence of the drastic appearance variation, [54] further introduces a novel multi-center metric learning framework that models the latent views from the vehicle’s visual appearance directly without the need for extra labels. notwithstanding the successes in this literature, all the previously mentioned vehicle ReID methods learn real-valued continuous feature vectors and rely on the computation of Euclidean distance for retrieval. These approaches are not applicable in large-scale ReID settings where the database stores a significantly larger number of image items and the retrieval speed is of critical concern.

III. METHODOLOGY

In this paper, we denote mapping functions with calligraphic uppercase letters, such as \( \mathcal{F} \). We denote sets with boldface uppercase letters such as \( \mathbf{B} \). Images are denoted with italic uppercase letters such as \( I \). Specifically, the sign activation function is represented as \( \text{sign} \).

A. Problem Definition

Assume that \( \mathbf{X} = \{I_i\}_{i=1}^N \) is a training vehicle image set which contains \( N \) training instances. \( \mathbf{Q} = \{I_i\}_{i=1}^{N_0} \) and \( \mathbf{G} = \{I_i\}_{i=1}^{N_i} \) are the query set and gallery set, respectively. \( \mathbf{Y} = \{y_i\}_{i=1}^N \in [0, 1]^C \) is the corresponding label set and \( C \) is the number of identities. The goal of deep hash learning for vehicle re-identification is to learn a non-linear hash function \( \mathcal{H} : I \mapsto \mathbf{b} \in \{0, 1\}^K \) which takes as input an image \( I \) and map it into a K-bit binary hash code vector \( \mathbf{f} \) such that the visual similarity in the raw pixel space is preserved for each vehicle image in the hamming space. The whole learning-based hash function could be formulated as:

\[
\mathbf{b} = \mathcal{H}(I) = \text{sign}(\mathcal{F}_{\text{hash}}(I))
\]

where \( \mathcal{F}_{\text{hash}} \) stands for the non-linear deep neural network responsible for generating hash codes. \( I \) is an image. \( \text{sign}(v) \) returns 1 if \( v \geq 0 \) and returns -1 if \( v < 0 \).

IV. PROPOSED MODEL

In this section, we elaborate on the detailed implementation of our proposed model - ‘DVHN’. We first give a brief introduction on the network architecture, followed by the metric learning method that is adopted. Then, we illustrate the
proposed discrete hashing learning module. Then, we introduce the optimization method adopted to optimize the whole framework. At last, we briefly introduce the training process and retrieval process.

### A. Hashing Layer

We adopt ResNet-50 [55] as the backbone feature extraction network, which is denoted as $F_{θ}$. The feature output vector $f_i$ for an image $I_i$ from $F_{θ}$ is denoted as:

$$f_i = F_{θ}(I_i)$$

To generate a binary hashing code of diversified lengths, we propose to devise a hashing layer on top of the backbone network. Concretely, following the global average pooling layer, we add a fully connected layer taking as input a feature vector $f$ of $M$ dim and output $K$-dim continuous hash vector $h$:

$$h = F_{θ}(f) = fW_1^T + b$$

where $W_1 \in \mathbb{R}^{M \times K}$ and $b \in \mathbb{R}^K$ denotes the weight and bias parameters while $K$ is the length of the Hamming hashing code. Note that, unlike previous deep hashing methods which adopt sigmoid or tanh activation function to squash the output into approximate binary codes, we propose a Quantization Loss to enforce the output to be binary, which is elaborated in Sec: IV-C.

### B. Similarity-Preserving Feature Learning

We organize the input vehicle into triplet samples and formulate the hash problem as a similarity-preserving problem. Concretely, inspired by [14], we form input batches by first randomly sampling $P$ vehicle identities, then, we randomly sample $K_1$ instances for each vehicle. Thus, we get a batch of $P \times K_1$ images in total. For every image $I_i$ in the batch, we will use it as an anchor image and then find an image $I_i^+$ which is the hardest positive sample in the batch. Analogously, we can select the hardest negative sample for the anchor $I_i$ in the batch, which is the image belonging to another vehicle but closest to the anchor $I_i$ in the batch according to a predefined distance metric $D$. In doing so, a training set $T_1 = \{(I_i, I_i^+, I_i^-)\}_{i=1}^{P \times K_1}$ is derived. For each image $I_i$ the corresponding continuous hash vector $h$ is computed with the hash function $H$:

$$h_i = H(I_i) = F_{θ}(F_{θ}(I_i))$$

where $h_i$ is the continuous hash vector for image $I_i$ and $F_{θ}$ denotes the backbone non-linear convolutional neural network structure. $F_{θ}$ denotes the fully connected feed-forward neural network (hash layer) for hash code generation. Since the goal is to preserve the similarity of the images in the hamming space, it is natural to propose the following constraint:

$$D_H(h_i, h_i^+) < D_H(h_i, h_i^-)$$

where $D_H$ represents the hamming distance metric. The intuition is that the hamming distance between the anchor and the hardest positive sample should be smaller than the negative counterpart. We observe a linear relation between the Hamming distance and the inner product of two vectors:

$$D_H(h_i, h_j) = \frac{1}{2}(β - \langle h_i, h_j \rangle)$$

Thus, in this paper, we adopt inner product as a nice surrogate for Hamming distance calculation.

In this way, we can finally formulate the triplet-based similarity preserving loss as:

$$L_{Triplet}(h) = \sum_{i \in T_1} \left[ \alpha + D_H(h_i, h_i^+) - D_H(h_i, h_i^-) \right]$$

$$= \sum_{i=1}^{P} \sum_{a=1}^{K} \left[ \alpha + \max_{p=1..K} \| e_i^a - e_i^p \|_2 \right]$$

$$- \min_{a=1..K} \left[ \| e_i^a - e_i^n \|_2 \right]$$

where $P$ denotes the number of vehicle identities in the batch and $K$ stands for the number of instances for each identity in the batch. $e_i^a$ denotes the anchor’s vector belong to identity $i$. $α$ is the margin parameter for the triplet loss.

On top of the triplet loss $L_{Triplet}$, to learn more discriminative features, we further add an Identity Loss on the feature output $f$. To this end, another fully connected layer $F_{θ_i}$ of $N_C$ output logits is added after the $f$ as shown in Fig. 1. $N_C$ is the number of vehicle identities for the training images. By regarding each vehicle identity as a class, we can formulate the identity loss $L_{Identity}$ as:

$$L_{Identity}(f) = -\sum_{i=1}^{P \times K} \log \frac{e_{w_i}^T f}{\sum_{k=1}^{N_C} e_{w_i}^T f_k}$$

where $W_k$ is the weight vector for identity $k$, $P \times K_1$ is the number of images in the batch and $N_C$ is the number of identities in the batch.

### C. Discrete Hashing Learning

The core assumption is that the learned binary Hamming codes should preserve the label information. Specifically, for each image $I_i$, the binary Hamming code $b_i$ is obtained by using the $sgn$ activation function $sgn(x) = \begin{cases} 1, x > 0 \\ -1, x \leq 0 \end{cases}$ on the continuous hash vector $h_i$ as:

$$b_i = sgn(h_i) = sgn(H(I_i))$$

Similar to [56], we adopt a simple linear classifier to reconstruct the input ground-truth labels with the binary codes as:

$$y_i = W_b^T b_i$$

where $y_i \in \{0, 1\}^C$, $C$ is the number of identities, $W_b$ is the weight of the classifier for the binary codes $b$. Then, we could formulate the discrete hashing loss as:

$$L_{Quant}(b) = \mu \sum_{i=1}^{N} \| y_i - W_b^T b_i \|_2^2 + \nu \| W_b \|_F^2$$
where \( \| \cdot \|_2 \) is the \( l_2 \) norm of a vector while \( \| \cdot \|_F \) is the Frobenius norm of a specific matrix.

**D. Alternative Optimization**

We formulate the final total loss function as:
\[
L_{Total} = \lambda L_{Triplet}(h) + \sigma L_{Identity}(f) + L_{Quant}(b) \quad (10)
\]
where \( \lambda, \beta, \) and \( \sigma \) are three predefined coefficients for controlling the influence of each individual loss. One might ponder the possibility of optimizing the Eq. 10 with standardized stochastic gradient descent. Nonetheless, the \( \text{sign} \) function is non-differentiable which cannot be directly optimized by back-propagating the gradients. Thus, the third term \( L_{Quant}(b) \) could not be optimized in a same way as the first two terms. We, thus, unfold the last term in the Eq. 10:
\[
L_{Total} = \lambda L_{Triplet}(h) + \sigma L_{Identity}(f) + \mu \sum_{i=1}^{N} \| y_i - W_h^T b_i \|_2^2 + \nu \| W_h \|_F^2 \quad (11)
\]

To circumvent the vanishing gradient problem introduced by \( \text{sign} \), some previous works adopt \( \tanh \) or \( \text{sigmoid} \) activation function to generate continuous outputs which are the relaxation of the binary Hamming codes in the training stage. In the testing stage, \( \text{sign} \) activation function is adopted to obtain the corresponding binary Hamming hashing codes. However, such an optimization scheme is optimizing a relaxed surrogate of the original target and will result in learning sub-optimal Hamming hashing codes owing to the sizable Quantization Error [27].

In this regard, we propose to adopt a novel Alternative Optimization scheme that considers the discrete nature of binary codes by constraining the outputs from the deep neural network to be binary codes directly. Similar to [56] and [57], we introduce an auxiliary variable and derive an approximation of Eq. 11 as:
\[
L_{Total} = L_{Triplet}(h) + L_{Identity}(f) + \mu \sum_{i=1}^{N} \| y_i - W_h^T b_i \|_2^2 + \nu \| W_h \|_F^2 \\
\text{s.t.} \quad b_i = \text{sgn}(h_i), \quad b_i \in \mathbb{R}^{K \times 1}, \quad (i = 1, \ldots, N) \quad (12)
\]

where \( h = f_{hash}(f) \) is the continuous hashing vector, \( f \) is the feature output from the deep convolutional neural network and \( b \) is the binarized Hamming hashing code. The above equation is a constrained optimization problem, which could be solved with the Lagrange Multiplier Method as:
\[
L_{Total} = L_{Triplet}(h) + L_{Identity}(f) + \mu \sum_{i=1}^{N} \| y_i - W_h^T b_i \|_2^2 + \nu \| W_h \|_F^2 \\
+ \eta \sum_{i=1}^{N} \| b_i - \text{sgn}(h_i) \|_2^2 \\
\text{s.t.} \quad b_i \in \{-1, 1\}^K, \quad b_i \in \mathbb{R}^{K \times 1}, \quad (i = 1, \ldots, N) \quad (13)
\]
The last term can also be viewed as a constraint that measures the quantization error between the continuous hashing feature vector and its binarized Hamming code. Inspired by [56], we decouple the above optimization target Eq. 13 into two sub-optimization problems and solve them in an alternative fashion.

First, we fix \( b \) and \( W \), the optimization target is converted to

\[
L = \lambda L_{Triplet}(h) + \sigma L_{Identity}(f) + \eta \sum_{i=1}^{N} \| b_i - h_i \|_2^2
\]  

(14)

Then, the parameters of the backbone \( \theta \), hash layer \( \theta_1 \), and the identity loss \( \theta_2 \) could be updated with standard Stochastic Gradient Descent and Back Propagation algorithm.

Secondly, we fix the parameters \( \theta, \theta_1, \theta_2 \) and \( b_i \) and the problem is converted into:

\[
L = \mu \sum_{i=1}^{N} \| y_i - W_h^T b_i \|_2^2 + \nu \| W_h \|_F^2
\]  

(15)

It is obvious to note that this is a Least Squares problem and has a closed-form solution as:

\[
W_h = \left( BB^T + \frac{\nu}{\mu} I \right)^{-1} B^T Y
\]  

(16)

where \( B = \{ b_i \}_{i=1}^{N} \in \{-1, 1\}^{K \times N} \) and \( Y = \{ y_i \}_{i=1}^{N} \in \mathbb{R}^{C \times N} \).

Finally, after solving \( W_h \), when we fix the parameters \( \theta, \theta_1, \theta_2 \) and \( W_h \), we obtain the final objective as:

\[
L = \mu \sum_{i=1}^{N} \| y_i - W_h^T b_i \|_2^2 + \eta \sum_{i=1}^{N} \| b_i - h_i \|_2^2
\]  

s.t. \( b_i \in \{-1, 1\}^K \), \( (i = 1, \ldots, N) \)  

(17)

Inspired by [56], we adopt the discrete cyclic coordinate descent method to solve \( B \) row by row in an iterative fashion as:

\[
\min_B \| W_h^T B \|_F^2 - 2 Tr(P), \quad \text{s.t.} \ B \in \{-1, 1\}^{K \times N}
\]  

(18)

where \( P = W_h Y + \frac{\eta}{\mu} H \).

E. Overall Training Algorithm

The overall training algorithm for DVHN is illustrated in Algo. 1. We have a training image set \( T \) and the corresponding label set \( Y \). The initial learning rate is set to \( \epsilon \) and we set the three hyper-parameters \( \lambda, \beta, \sigma \) for balancing the total loss in Eq. 10 and the margin parameter \( \alpha \). The overall number of training iterations \( T \) is also set to a specific number. We first initialize the parameters \( \theta \) of the ResNet-50 backbone neural network with the pre-trained weights on ImageNet. The parameters \( \theta_1 \) for the hashing layer and the parameters \( \theta_2 \) for the classifier are initialized randomly, and the counter \( t \) for counting the number of iterations is initialized to 0 at the very beginning.

When the model is not converged and the iteration \( t \) is less than the total number of iterations \( T \), we perform the following operations. We first increase the iteration counter \( t \) by one. Then perform the following operations for 100 steps.

First, we freeze the parameters for the discrete hashing module \( W \) and the binary codes \( B \). Then, we construct a batch of images \( I_b \) which contains \( P \times K \) images in total. \( P \) is the number of vehicle identities, and \( K \) is the number of images randomly selected for a specific vehicle identity. The corresponding label set \( Y_b \) is also constructed. Later, we obtain the triplet loss \( L_{Triplet} \) and the identity loss \( L_{Identity} \) as suggested in the Algo. 1. Then, we adopt backpropagation to obtain the gradients \( \theta_k \) for \( \theta, \theta_1, \theta_2 \). Finally, we update the parameters with the learning rate \( \alpha \). After 100 steps of the above operations, we then freeze the parameters \( \theta, \theta_1, \theta_2 \), and update the parameter for the discrete hashing module \( W_h \) with Eq. 15. Then, we freeze \( \theta, \theta_1, \theta_2, W_h \), and update the binary codes for all the images \( B \) row by row through Eq. 18.

F. Retrieval Process

In this section, we elaborate on how to perform efficient vehicle re-identification given the trained model. Generally, we are given a query image \( q \) and a gallery image set \( G \). For each query image \( q_k \), the task is to identify the images in \( G \) belonging to the same vehicle. In the testing phase, for each query image \( q_k \), we obtain the discrete binary code vector \( b_k^q \) through:

\[
b_k^q = \text{sign}(F_{\theta_1}(F_0(I_k^q))
\]  

(19)
Subsequently, in the same fashion, for all the images in \( G = \{g_k \}_{k=1}^{N_g} \), we obtain a set of binary code vectors \( H^p = \{b^p_k \}_{k=1}^{N_g} \). Then, we can rank the gallery code vectors through their hamming distance with respect to the query image code vector. We note that there is a linear relationship between inner product \( \langle \cdot, \cdot \rangle \) and hamming distance \( D_H \), which is stated as:

\[
D_H(b_i, b_j) = \frac{1}{2}(K - \langle b_i, b_j \rangle)
\]

where \( K \) is the bit length of the binary hashing code \( b \). The above equation makes the inner product a nice surrogate for hamming distance computing. In the later stage, we compute the hamming distance between \( b^p_k \) and all the gallery hash codes in \( G \). Then, we sort the distances in increasing order. Finally, we rank the images in \( G \) according to the sorted hamming distances.

V. EXPERIMENTS

In this section, we evaluate our method against the state of the arts on standardized public datasets. We first give a brief introduction on the implementation details of our architecture, information on the benchmark datasets, and the evaluation protocols. Then, we present detailed analyses of the experimental results of our method and the compared state-of-the-arts.

A. Implementation Details

We implement our model with pytorch [58]. The detailed framework is illustrated in Fig. 1. Specifically, after the final \( f_{avg} \) layer of resnet50, we add a fully connected layer of \( K \) output logits where \( K \) denotes the hash code length. For the classification stream, we added another fully connected layer of \( N_C \) output logits, \( N_C \) denoting the number of classes. These two additional layers are initialized with “normal” initialization, the mean being 0 and the standard deviation being 0.01. In the training process, the number of identities \( P \) is set to 16, and the number of instances is 6, resulting in a batch of 96 images. During training, the parameters are optimized with “amsgrad” optimizer. The initial learning rate \( \epsilon \) is set to \( 3e-4 \), the weight decay to \( 5e-4 \), the beta1 to 0.9, and the beta2 to 0.99. The margin parameter \( \alpha \) in Eq. 5 is set to 0.3. For the coefficients controlling the influence of each loss \( (\lambda, \beta, \sigma) \), we empirically set all of them to 1.

B. Datasets and Evaluation Protocols

1) Datasets: We conduct experiments on two widely-studied public datasets: VehicleID [1] and Ve Ri [27]. VehicleID is a large-scale vehicle re-identification dataset containing 221763 images of 26267 vehicle identities in total. Among all the images, 110,178 images of 13,134 vehicles are selected for training. For testing, three subsets which contain 800, 1600, and 2400 vehicles are extracted from the testing dataset and are denoted as 800, 1600, 2400, respectively. We empirically evaluated the performances of our model on 800 and 1600 scenarios. Specifically, for 800, the query set contains 5693 images of 800 vehicle identities, and the gallery dataset contains 800 images in total, one for each vehicle identity. In a similar fashion, for the 1600 scenario, the query set comprises 11777 images of 1600 cars, while the gallery set contains 1600 images of 1600 identities. Ve Ri is a real-world urban surveillance vehicle dataset containing over 50000 images of 776 vehicle identities, among which 37781 images of 576 vehicles are designated as the training set. For testing, 1678 images of 200 vehicles are designated as the query set, while the gallery contains 11579 for the same 200 vehicles.

2) Evaluation Protocols: We evaluate our model on two widely-adopted metrics: cumulative matching characteristics (CMC) and mean average precision (mAP). The CMC score computes the probability that a query image appears in a different-sized retrieval list. However, it only counts the first match, making it unsuitable in a scenario where the gallery contains more than one image for each query. Thus, we also employ mAP, which, essentially, takes into consideration both the precision and recall of the retrieved results. Specifically, as suggested in [59] for each query image, we compute the area under the Precision-Recall curve, which is given by:

\[
AP = \sum_{k=1}^{N} precision_k \cdot (recall_k - recall_{k-1})
\]

where \( k \in \{1, 2, 3, \ldots, N\} \) and \( N \) is the total number of images in the database. And mAP is the mean value of all the APs of all the queries. In this way, for query images of multiple positive counterparts in the database, the recall of the retrieval list is also considered.

C. Comparisons With State-of-the-Arts

In this section, we compare the results of our proposed DHN and the state-of-the-art deep hashing methods. Specifically, the competing methods could be divided into two categories: unsupervised hash methods and supervised hash methods. For the unsupervised methods, we include SH [16] and ITQ [17]. Note that these two methods are reliant upon extra feature extractors to obtain feature vectors for the dataset. Generally, we use pre-trained deep neural networks to perform this task. Specifically, ImageNet-pretrained AlexNet is adopted in this paper. For the supervised setting, we further incorporate DSH [60] which was among the first works attempting to learn discrete hashing codes with deep convolutional neural networks. In addition, we incorporate other recent methods for detailed comparisons and analyses including DHN [26], HashNet [11], DCH [34], DP N [24]. Note that we adopt original source codes for SH, ITQ, DCH and DHN. For other methods, we implement them with pytorch. We first provide analysis for the results of all the models with 2048 hash bits. It is rather evident from Tab.I and Tab. II that traditional unsupervised methods like SH and ITQ exhibit significantly poorer performance across two datasets. Their mAP scores on Ve Ri are 1.57% and 5.14% when the hashing bit length equals 2048, respectively. On VehicleID (800) and VehicleID (1600), the performance is slightly better. Specifically, Sh achieves mAPs of 17.71% and 13.83% while ITQ reaches 26.26% and 22.23% for 2048 bits on these two datasets. The unsatisfactory results could in large part be attributed to the
fact that unsupervised methods could not learn the discriminative features, leading to sub-optimal hash codes. Clearly, supervised methods achieve better results across two datasets for all the hashing bit lengths. Especially, HashNet and DPN achieve pronounced results on VehicleID (800), 66.61% and 64.65% for 2048 bits, respectively. Across other hashing bit lengths, they are also clear winners regarding their unsupervised counterparts into comparisons considering their ent hashing bit lengths. Specifically, we achieve 11.52%, 9.61%, 10.75%, and 11.05% performance gains against DPN in 256, 512, 1024, and 2048 hash bits. It is also easy to note that HashNet can achieve comparable performance from Rank@6 to Rank@20 when hash bit length equals 256 and 512. However, in terms of the vehicle re-identification task, it is of crucial importance that the relative matching images appear in the front of the retrieval list. In terms of mAP, as depicted in the above part of Fig. 2, DVHN beats all the other competing methods with notable margins. It is also easy to spot that with the number of bits increasing, our proposed model achieves growing mAP scores from 71.61% to 76.81%. The performance gain could be attributed to the fact that the model captures more information with a comparatively larger hash bit length.

2) VehicleID (1600): For test performance on this scenario, as demonstrated in Fig. 2, a similar trend is spotted for both CMC and mAP. We achieve 57.06%, 60.41%, 63.26%, and 64.11% of mAP for HashNet and 13.58% for DPN on 2048 bits. In terms of VeRi, poorer performance could be spotted, 29.30% mAP for HashNet and 13.58% for DPN on 2048 bits. The degraded performance could result from the small discrepancy of images from the same vehicle. Unsurprisingly, our proposed DVHN is a clear winner against all the competing methods across all the compared bit lengths. Concretely, we achieve the state-of-the-art results with pronounced margins of 11.05% and 35.46% in terms of Rank@1 on VehicleID and VeRi on 2048 bits, respectively. For 256 bits, we obtain 11.54% and 30.93% performance improvements of Rank@1. In terms of mAP, for 2048 bits, we achieve 76.82%, 72.72% and 56.57% on VehicleID (800), VehicleID (1600) and VeRi, outperforming the second-best results by 10.21%, 11.30% and 32.72%. The superior performances collectively and evidently demonstrated the advantages of the joint hash and classification stream design, which enables the model to learn more discriminative features.

We further provide the top-K accuracy results (CMC curves) of different hash bit lengths on VehicleID (800), VehicleID (1600) and VeRi in Fig. 2.

1) VehicleID (800): As depicted in Fig. 2, we compare the CMC results of our model DVHN with other state-of-the-art deep supervised hash methods across four different hashing bit lengths. Note that we do not include two unsupervised counterparts into comparisons considering their unsatisfactory performance in this domain. Evidently, DVHN achieves notable results especially from Rank@1 to Rank@5.

Table II

| Datasets   | VehicleID (1600) | VehicleID (800) |
|------------|------------------|-----------------|
| Methods    | VehicleID (1600) | VehicleID (800) |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |
| DSH[60]    | 256              | 512             |
|            | 30.56            | 32.53           |
|            | 34.22            | 33.62           |

2) VehicleID (1600): For test performance on this scenario, as demonstrated in Fig. 2, a similar trend is spotted for both CMC and mAP. We achieve 57.06%, 60.41%, 63.26%, and 64.11% of mAP for HashNet and 13.58% for DPN on 2048 bits. In terms of mAP, as depicted in the above part of Fig. 2, DVHN beats all the other competing methods with notable margins. Note that, when compared with performance on VehicleID(800), the model experiences moderate performance declines for all hash bits. For instance, the Rank@1 accuracy drops by over 4% for hash bit 256 while the mAP drops by near 5%. The degraded performance is mainly due to the introduced difficulty of the larger gallery set.

3) VeRi: As is manifested in Fig. 2, DVHN exhibits consistent advantages against all the competing methods across all the evaluation metrics. Specifically, as shown in Fig. 2, DVHN outperforms HashNet by large margins, consistently. We achieve 80.10%, 85.81%, 87.01% and 88.62% of mAP for Rank@1 for 256, 512, 1024, and 2048 hash bits, respectively. In regard to mAP, we outperform all the candidate methods by large margins in all testing scenarios. Note that, when compared with performance on VehicleID(800), the model experiences moderate performance declines for all hash bits. For instance, the Rank@1 accuracy drops by over 4% for hash bit 256 while the mAP drops by near 5%. The degraded performance is mainly due to the introduced difficulty of the larger gallery set.

D. Empirical Analysis

First, we plot the convergence curves for our model across four hashing bits in Fig. 3.
Second, we conduct a detailed ablation study to demonstrate the effectiveness of each component of our proposed DVHN. To effectively evaluate the importance of Identity Loss ($L_{Identity}$), triplet-based hash loss ($L_{Triplet}$), and the proposed discrete learning loss ($L_{Quant}$), we empirically test the performance of our model on VehicleID (800) and VeRi when stripped of each component. We denote the model of all the components as Full Model, the model without triplet loss as W/O Triplet, the model without Identity Loss as W/O Identity and, finally, the model without the discrete hashing learning as W/O Quant. Note that all the experiments were conducted on 2048 hash bits. As demonstrated in Fig. 5(a) and Fig. 5(b), when the discrete learning loss is moved from the model, we experience slight performance declines, 3.46% and 3.5% for Rank@1 and mAP on VehicleID (800), and 2.80% and 3.77% for Rank@1 and mAP on VeRi. This evidences the validity of the discrete hashing learning module for the binary Hamming hashing codes to preserve the class information.
When deprived of the identity loss, the model experiences conspicuous performance decreases. On VehicleID (800), the Rank@1 accuracy drops by 40.84% while the mAP score drops by 35.31%. On VeRi, the performance deterioration is even more severe, 75.11% for Rank@1 and 52.84% for mAP. The declines have demonstrated the importance of identity loss in learning more robust and discriminative features. Finally, when looking at the performance of W/O Triplet, we could also spot notable performance declines against the Full Model. W/O Triplet achieves 57.47% of Rank@1 accuracy and 67.32% mAP, 10.19% and 9.45% smaller than its full model counterpart on VehicleID (800). On VeRi, a similar trend can also be spotted when the model is deprived of $L_{Triplet}$. W/O Triplet achieves 78.84% of Rank@1 and 49.23% of mAP, 9.78% and 12.79% lower than the Full Model counterpart. The deterioration is largely due to the effectiveness of batch-hard triplet loss in tackling the vehicle re-identification problem, where the intra-class variation is sometimes even larger than the inter-class variation. Finally, as elaborated above, all the proposed components are essential to the final performance of the proposed final model.

E. Retrieval Efficiency Comparison

The time cost of a Vehicle ReID system generally is comprised of two separate parts: feature extraction cost and similarity calculation cost. As is pointed out by [61], the cost of the feature extraction actually heavily depends on the backbone network used. Thus, we focus on comparing the large-scale similarity calculation time costs of our method and the real-valued vehicle re-identification counterparts. Meanwhile, we also compare the storage costs of our Hamming hashing re-identification framework with the real-valued ones.

Specifically, we illustrate the results on VehicleID (1600) and VeRi in Fig. 4, which have (11777, 1600), (1678, 11579) (query, gallery) images, respectively. In Fig. 4(a) and Fig. 4(c), the ‘Float-Euclidean’ denotes the query time for Euclidean distance-based real-valued vehicle re-identification methods while ‘Binary-Hamming’ denotes the query time for our binary hashing method with Hamming distance calculation. For Fig. 4(b) and Fig. 4(d), it demonstrates the comparison of storage cost on VehicleID (1600) and VeRi for real-valued
methods and our binary Hamming codes based method. It is obvious that binary codes could save tremendous memory compared with their real-valued counterparts.

F. Qualitative Analysis of the Retrieved Results

In Fig. 6, we visualize the retrieval results for different hash bit lengths of our model in VeRi. Note that, to demonstrate the retrieval efficiency, we take the original gallery set as a query set so that for each query image, there exist several images in the gallery set. In the picture above, we adopt the same query images for four testing scenarios where the hash bit length equals 256, 512, 1024, and 2048, respectively. The right part visualizes the retrieval list for the query image. The image enclosed by a blue box means it is from the same vehicle as the query image. Generally, for a re-identification model, it would be desirable for the matching images to appear in the front of the retrieval list.

VI. Conclusion

In this paper, we introduce the first deep hashing-based framework for efficient large-scale vehicle re-identification. Motivated by the deep hashing community which adopts binary codes for efficient image retrieval, we make the first attempt to adopt a binary Hamming hashing scheme for large-scale vehicle re-identification, which could save tremendous memory and is much faster in terms of retrieval speed compared to traditional real-valued re-identification methods. Specifically, for similarity-preserving learning, we adopt a triplet loss and an on-the-fly hard triplet generation module. Meanwhile, we further add an identity loss to learn more robust and discriminative features. For binary code learning, we propose a discrete hashing learning scheme that is based on the assumption that the binary codes should be ideal for classification. To overcome the vanishing gradient problem (the sign function for generating binary codes is not differentiable), we adopt an alternative optimization method for optimizing the overall framework. We conduct extensive experiments and the results have demonstrated that our framework could generate discriminative binary codes for efficient vehicle re-identification, surpassing the state-of-the-art deep hashing methods with large margins.

REFERENCES

[1] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang, “Deep relative distance learning: Tell the difference between similar vehicles,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2167–2175.
[2] W. Liu, X. Liu, H. Ma, and P. Cheng, “Beyond human-level license plate super-resolution with progressive vehicle search and domain priori GAN,” in Proc. 25th ACM Int. Conf. Multimedia, Oct. 2017, pp. 1618–1626.
[3] T. Yan, Y. Tian, Y. Wang, W. Zeng, and T. Huang, “Exploiting multi-grain ranking constraints for precisely searching visually-similar vehicles,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 562–570.
[4] H. Guo, C. Zhao, Z. Liu, J. Wang, and H. Lu, “Learning coarse-to-fine structured feature embedding for vehicle re-identification,” in Proc. AAAI Conf. Artif. Intell., vol. 32, no. 1, 2018, pp. 6853–6860.
[5] Y. Lou, Y. Bai, J. Liu, S. Wang, and L. Duan, “Embedding adversarial learning for vehicle re-identification,” IEEE Trans. Image Process., vol. 28, no. 8, pp. 3794–3807, Aug. 2019.
[6] D. Meng et al., “Parseing-based view-aware embedding network for vehicle re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 7103–7112.

[7] R. Chu, Y. Sun, Y. Li, Z. Liu, C. Zhang, and Y. Wei, “Vehicle re-identification with viewpoint-aware metric learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 8282–8291.

[8] K. Lin, H. Yang, J. Hsiao, and C. Chen, “Deep learning of binary hash codes for fast image retrieval,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognition Workshops (CVPRW), Jun. 2015, pp. 27–35.

[9] F. Cakir, K. He, S. A. Bargal, and S. Schroff, “Hashing with mutual information,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 10, pp. 2424–2437, Oct. 2019.

[10] Y. Cao, M. Long, J. Wang, and H. Zhu, “Correlation autoencoder hashing for supervised cross-modal search,” in Proc. ACM Int. Conf. Multimedia Retr., Jun. 2016, pp. 197–204.

[11] Z. Cao, M. Long, J. Wang, and P. S. Yu, “HashNet: Deep learning to hash by continuation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5608–5617.

[12] E.-J. Ong and M. Bober, “Improved Hamming distance search using variable length hashing,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2000–2008.

[13] H. Lai, Y. Pan, Y. Liu, and S. Yan, “Simultaneous feature learning and hash coding with deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3270–3278.

[14] A. Hermans, L. Beyer, and B. Leibe, “In defense of the triplet loss for metric learning,” in European Conf. Comput. Vis. (ECCV), 2017, pp. 334–350.

[15] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala, “Locality-preserving hashing in multidimensional spaces,” in Proc. 29th Annu. ACM Symp. Theory Comput. (STOC), 1997, pp. 618–625.

[16] Y. Weiss et al., “Spectral hashing,” in Proc. NIPS, 2008, vol. 1, no. 2, p. 4.

[17] T. Jebara, T. Zhang, and J. Shawe-Taylor, “Kernelized hashing,” in Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI), 2008, pp. 287–294.

[18] T. Jebara, T. Zhang, and J. Shawe-Taylor, “Kernelized hashing,” in Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI), 2008, pp. 287–294.

[19] Z. Tang et al., “PAMTRI: Pose-aware multi-task learning for vehicle re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2019, pp. 394–400.

[20] Y. Zhao et al., “Deep multi-view re-identification for vehicle re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2019, pp. 394–400.

[21] R.-C. Tu, X.-L. Mao, and W. Wei, “MLS3RDHU: Deep unsupervised hashing via manifold based local semantic similarity structure reconstructing,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 3466–3472.

[22] Z. Qiu, Q. Su, Z. Ou, J. Yu, and C. Chen, “Unsupervised hashing with contrastive information bottleneck,” 2021, arXiv:2105.06138.

[23] X. Luo et al., “CIMON: Towards high-quality hash codes,” 2020, arXiv:2010.07804.

[24] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, “Supervised hashing for image retrieval via image representation learning,” in Proc. AAAI Conf. Artif. Intell., vol. 28, no. 1, 2014, pp. 2156–2162.

[25] Y. Cao, M. Long, B. Liu, and J. Wang, “Deep Cauchy hashing for Hamming space retrieval,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1229–1237.

[26] D. Wu, Q. Dai, J. Liu, B. Li, and W. Wang, “Deep incremental hashing network for efficient image retrieval,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 9069–9077.

[27] X. Liu, W. Liu, H. Ma, and H. Fu, “Large-scale vehicle re-identification in urban surveillance videos,” in Proc. IEEE Int. Conf. Multimedia Expo (ICME), Jul. 2016, pp. 1–6.

[28] Z. Wang et al., “Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 379–387.

[29] X. Liu, W. Liu, T. Mei, and H. Ma, “PROVID: Progressive and multimodal vehicle reidentification for large-scale urban surveillance,” IEEE Trans. Multimedia, vol. 20, no. 3, pp. 645–658, Mar. 2018.

[30] Y. Bai et al., “Disentangled feature learning network for vehicle re-identification,” in Proc. IJCAI, 2020, pp. 474–480.

[31] C. Chen, G. Yao, C. Wang, S. Goudos, and S. Wan, “Enhancing the robustness of object detection via 6G vehicular edge computing,” Digit. Commun. Netw., vol. 8, no. 6, pp. 923–931, Dec. 2022.

[32] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, “An edge traffic flow detection scheme based on deep learning in an intelligent transportation system,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 3, pp. 1840–1852, Mar. 2021.

[33] S. Liu, J. Yu, X. Deng, and S. Wan, “FedCPP: An efficient-computation federated learning approach for vehicular edge computing in 6G communication networks,” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 2, pp. 1616–1629, Feb. 2022.

[34] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, “Data-driven intelligent transportation systems: A survey,” IEEE Trans. Intell. Transp. Syst., vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[35] Z. Yu, C. Licia, W. Ouri, and Y. Hai, “Urban computing: Concepts, methodologies, and applications,” ACM Trans. Intell. Syst. Technol., vol. 5, no. 3, pp. 1–55, Sep. 2014.

[36] B. He, J. Li, Y. Zhao, and Y. Tian, “Part-regularized near-duplicate vehicle re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3977–4005.

[37] Z. Zheng, T. Ruan, Y. Wei, Y. Yang, and T. Mei, “VehicleNet: Learning robust visual representation for vehicle re-identification,” IEEE Trans. Multimedia, vol. 23, pp. 2683–2693, 2021.

[38] M. Jiang et al., “Robust vehicle re-identification via rigid structure prior,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2021, pp. 4026–4033.

[39] J. Li, Y. Cong, L. Zhou, Z. Tian, and J. Qiu, “Super-resolution-based part collaboration network for vehicle re-identification,” in Proc. World Wide Web, 2022, pp. 1–20.

[40] Y. Zhou and L. Shao, “Aware attentive multi-view inference for vehicle re-identification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6489–6498.

[41] Z. Tang et al., “PAMTRI: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 211–220.

[42] J. Zhu et al., “Vehicle re-identification using quadruple directional deep learning features,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 1, pp. 410–420, Jan. 2020.

[43] X. Chen, H. Sui, J. Fang, W. Feng, and M. Zhou, “Vehicle re-identification using distance-based global and partial multi-regional feature learning,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 2, pp. 1276–1286, Feb. 2021.

[44] X. Liu, W. Liu, J. Zheng, C. Yan, and T. Mei, “Beyond the parts: Learning multi-view cross-part correlation for vehicle re-identification,” in Proc. 28th ACM Int. Conf. Multimedia, Oct. 2020, pp. 907–915.
[54] Y. Jin, C. Li, Y. Li, P. Peng, and G. A. Giannopoulos, “Model latent views with multi-center metric learning for vehicle re-identification,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 3, pp. 1919–1931, Mar. 2021.

[55] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[56] Q. Li, Z. Sun, R. He, and T. Tan, “Deep supervised discrete hashing,” 2017, arXiv:1705.10999.

[57] X. Wang, Y. Shi, and K. M. Kitani, “Deep supervised hashing with triplet labels,” in Proc. Asian Conf. Comput. Vis. Taipei, Taiwan: Springer, 2016, pp. 70–84.

[58] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” 2019, arXiv:1912.01703.

[59] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1116–1124.

[60] H. Liu, R. Wang, S. Shan, and X. Chen, “Deep supervised hashing for fast image retrieval,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2064–2072.

[61] J. Chen et al., “Deep local binary coding for person re-identification by delving into the details,” in Proc. 28th ACM Int. Conf. Multimedia, Oct. 2020, pp. 3034–3043.

Yongbiao Chen received the bachelor’s degree in software engineering from Northwestern Polytechnical University, Xi’an, China. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include the application of machine learning and computer vision, with a primary focus on developing efficient algorithms for large-scale information retrieval.

Sheng Zhang received the bachelor’s degree in software engineering from Shanghai Jiao Tong University, Shanghai, China, and the master’s degree from the Department of Computer Science, University of Southern California, Los Angeles, USA. His research interests include machine learning, deep learning, and the application of deep learning to address the challenges of big data issues.

Fangxin Liu received the Ph.D. degree from Shanghai Jiao Tong University in 2023. He is currently an Assistant Professor with Shanghai Jiao Tong University. His research interests include neural network acceleration (e.g., neural network compression technique and SW/HW co-design), in-memory computing, and brain-inspired neuromorphic computing.

Zhengwei Qi received the B.Eng. and M.Eng. degrees from Northwestern Polytechnical University, in 1999 and 2002, respectively, and the Ph.D. degree from Shanghai Jiao Tong University in 2005. He is a Professor with the School of Software, Shanghai Jiao Tong University. His research interests include program analysis, model checking, virtual machines, and distributed systems.

Chengcheng Wu received the B.Eng. degree in ECE from Shanghai Jiao Tong University, where he is currently pursuing the Ph.D. degree in computer science. His research interests include edge computing and heterogeneous computing.

Kaicheng Guo received the B.Eng. degree in electronic information engineering from Northwestern Polytechnical University in 2017. He is currently pursuing the Ph.D. degree in computer science with Shanghai Jiao Tong University. His research interests include machine learning systems and heterogeneous computing.