Vehicle Re-Identification by Deep Feature Embedding and Approximate Nearest Neighbors

Artur O. R. Franco[‡], Felipe F. Soares[*‡], Aloísio V. Lira Neto[§],
José A. F. de Macêdo [¶‡], Paulo A. L. Rego[*‡], Fernando A. C. Gomes[‡], and José G. R. Maia[†‡]

[*]Group of Computer Networks, Software Engineering and Systems (GREat)
[†]Virtual UFC Institute
[¶]Insight Data Science Lab
[‡]Federal University of Ceará (UFC)
[§]Brazilian Federal Highway Police
Email: {artur,felipefs}@alu.ufc.br, {carvalho, jose.macedo, pauloalr, gilvanm}@ufc.br, aloisio.lira@prf.gov.br

***Abstract*—Disorganized urban growth has led cities to chaos, which has brought countless challenges for their development in several sectors, such as traffic organization, public safety, and transportation. Vehicular re-identification (ReID) technologies have become increasingly important in this context since these allow to produce insights capable of benefiting many areas. As recently, methods frequently resorted to Deep Learning and Convolutional Neural Networks (CNNs), especially in the design of loss functions capable of improving the learning capacity of CNNs. Couple and triplet loss techniques have gained prominence, but their effectiveness depends on the mining of samples to converge properly. In this paper, we investigate a new simple approach for vehicle ReID by combining sample mining strategy and approximate nearest neighbor (ANN) method to improve retrieval quality. By relying only on relatively low-dimensional deep features, we were able to obtain state-of-the-art performance on the VeRi-776 dataset in terms of mAP, HIT@1, and HIT@5 metrics, but using relatively simple CNN and ANN methods, which are feasible in CPU for real-time scenarios.**

***Index Terms*—CNN, Deep Learning, Vehicle Re-Identification.**

I. Introduction

Object re-identification (ReID) is a prominent research field that recently attracted the attention of the computer vision community [1], [2]. ReID refers to the matching of different objects appearing in images captured by different cameras, i.e., ReID is an important aspect of the Multi-Target Multi-Camera (MTMC) tracking problem. Initially applied to people, ReID consists of detecting and re-identifying subjects. Person ReID can rely on many face recognition techniques developed and enhanced over the years [1]. Some of these have also been applied to vehicle re-identification [3].

In particular, vehicle ReID emerges as a potential key tool for building solutions to many problems faced in modern transportation, safety, catastrophe, and emergency management, and other aspects of urban center administration [3]. When applied to public safety, the vehicle ReID enables finding suspects or stolen vehicles [4] even with license plate adulteration where traditional automatic license plate recognition (ALPR) and part identification methods usually fail in this case [2], [5], [6]. This is because vehicle ReID methods rely mainly on holistic appearance, i.e., are capable of benefiting from discerning features like model, make, color, stickers, numbers, scratches, etc [7]–[9].

Practical vehicle ReID is a challenging task due to many complicating underlying aspects, such as multiple viewing angles, occlusion, weather, and the presence of near-identical instances of same make, model, and color [10]. That is, vehicles usually present high intra-class variance and low inter-class variance [3], [6], [9].

Recent advances in Deep Learning and Convolutional Neural Networks (CNNs) have led to significant improvements in the state-of-the-art of many fields, especially image, speech, text, and audio processing [11]. Remarkable CNN models introduced breakthroughs in face recognition and object detection, for example [1], [12], [13], while dramatically reducing the human effort in designing methods to acquire specific, problem-driven visual features from images [14]. Concerning ReID, deep neural networks are widely used to produce highly distinctive vector representations for objects, i.e., vector embeddings which are a key component of most vehicle ReID methods [5]. The typical dimension of vehicle vector embeddings used in state-of-the-art is increasing with time along with more complex neural network architectures in order to improve results on reference vehicle ReID datasets [2], [3], [5]–[7], [15], [16]. In addition, crafty design of loss functions and training strategies is a key aspect for obtaining a competitive ReID method [1], [10].

Triplet Loss [17] is a recurrent element in most similar works, either as the main method or as a variation of it [2], [3], [10]: the general hypothesis assumed is that the most discern embeddings obtained by adopting this loss function favors ReID retrieval quality improvements.

In this work, we propose a vehicle ReID system designed for public safety that relies on a CNN architecture to produce a relatively low-dimensional vector embedding with state-of-the-art performance. Experimental evaluation shows that a less accurate deep metric in terms of instance matching can lead to better ReID performance. Moreover, the adoption of Approximate Nearest Neighbors (ANN) methods can provide

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

significant quality improvements at low retrieval times. These are the main contributions of this paper:

- We describe a method and system for vehicle ReID.
- We adopt a simple, "lightweight" network for feature extraction with competitive results contrasting to state-of-the-art methods.
- Our experimental evaluation shows that ReID performance is not exactly determined by matching accuracy based on metric optimization in terms of the traditional triplet loss function.
- Our results show that ANN methods can improve ReID quality by a significant margin.

The remaining of this work is organized as follows. Section II discusses related work. Our methodology is described throughout Section III. Section IV is devoted to experimental evaluation, which has produced the results discussed in Section V. Finally, the conclusions are presented in Section VI.

II. Related Work

Similar to Person ReID, which relies on face recognition technology, Vehicle ReID may resort to existing vehicle recognition algorithms that are mostly based on ALPR methods. However, this approach has serious limitations [3]. ALPR algorithms provide the license plate text, which raises concerns about privacy [5]. Depending on the capture conditions, the license plate information is not sufficiently reliable to accurately identify a vehicle, as ALPR algorithms are subject to errors. Also, license plates may be purposely replaced by car thieves or malicious people. Finally, images in which the license plate does not appear or is obstructed become virtually useless even when these contain other discerning characteristics [2].

The rest of this section is devoted to a discussion about vehicle ReID concerning the available datasets, evaluation, and performance metrics, and the most representative methods found in our literature review.

A. Datasets for Vehicle ReID

PKU VehicleID is a large benchmark dataset proposed by Hongye Liu et al. [6], which contains 221,567 images from 26,328 vehicles captured from a video monitoring system operating under normal condition. There are about 8.42 images per vehicle. There is no standard division of the samples in the dataset for training and testing purposes. Consequently, there is a chance that results obtained from other researchers may present subtle but little to insignificant divergence.

VeRi-776 [7] has been the most used benchmark for vehicle ReID. The images from this dataset were collected from 20 non-overlapping cameras of video traffic in China with a variety of camera angles vehicle viewpoints, illuminations, and occlusions. consists of 776 identities for training and 200 identities for testing.

CityFlow [8] is a public dataset that is onlinefor the AI City Challenge carried out by NVIDIA at CVPR 2019. This dataset is the largest-scale in terms of spatial coverage and the number of cameras in an urban environment. It consists of more than 3 hours of synchronized HD videos from 40 cameras across 10 intersections, with the largest distance between two simultaneous cameras being 2.5 km. CityFlow contains more than 200K annotated bounding boxes of vehicles with different viewing angles, scenes, and models. There is some information in the dataset about camera geometry and calibration to be used in the spatiotemporal analysis. A subset from this dataset is available for the task of image-based vehicle re-identification: *CityFlow-ReID* dataset contains 56,277 images, being 36,935 for training and 18,290 for ReID based on 1,052 query images. A total of 333 vehicles is captured 35 cameras for training. Moreover, there are 333 vehicles not found in the training set, obtained from 5 different cameras, available for evaluation. Based on recent results and activity, the CityFlow-ReID dataset is more challenging than VeRi [3], and it is still growing, so this dataset may replace VeRi as the golden standard in the field of vehicle ReID.

Finally, VeRi-Wild [9] contains images captured from 174 cameras under different scenarios across one month (30x24h). These cameras are distributed in a large urban district of more than 200 km^2. A remarkable effort was carried out to build and annotate this database: there are 12 million images from 40,671 unique vehicles under different capture, traffic, lighting, backgrounds, resolutions, viewpoints, occlusion, and especially challenging weather conditions. Results published so far suggest this is the most challenging dataset so far.

B. Retrieval, Evaluation and Performance Metrics

The problem of vehicle ReID can be formalized as follows. Given an integer k and an input query image q of a vehicle identified by an integer vid_q, considering a large image collection S, return an ordered list of k images that more closely are from the same vehicle vid_q according to a given similarity metric (see Fig. 1). Considering performance evaluation, one must consider an integer Q, the number of query images used to retrieve ranked lists in the test dataset.

Let us consider a single query, where GTP is the total number of ground truth positive images for the query and $TPseen$ is 1 when the $i-th$ image found in response to that query is a true positive. So, *Average Precision* $AP(q)$ for a given query image q measures how accurate the returned list of images is. $AP(q)$ is defined as:

$$AP = \frac{1}{GTP}\sum_{i=1}^{k}\frac{TPseen}{i} \tag{1}$$

In its turn, the overall performance of re-identification for a dataset is usually expressed in terms of Mean Average Precision (mAP), which is defined as:

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{2}$$

Moreover, the image-to-track $HIT@1$ and $HIT@5$ metrics are also used in most recent publications [2], [3], [9], [16]. Many works are also adopting the Cumulative Match Curve (CMC) to describe the probability that the target vehicle ID vid_q appears in results of different sizes (k). CMC is more

Fig. 1. Vehicle ReID: given a query image q (at left, highlighted in blue) return k images of the same vehicle (correct retrievals are highlighted in green). Evaluation metrics should be sensitive to the position of good matches in the resulting ranked list: given the same query image in the last two rows (e), the very last row displays better performance since the wrong retrieval occurs later in the list. These are actual results from our TL-GNT (odd rows), and CL-NGT (even lines) approaches. The image aspect ratio is kept for a better understanding of how the results can vary. Best viewed in color.

frequently reported in publications adopting the VehicleID dataset. CMC can be computed from $gt(q,k)$, which is 1 when vid_q appears in the top-k results of the list

$$CMC = \frac{\sum_{q=1}^{Q} gt(q,k))}{Q} \tag{3}$$

C. Methods for Vehicle ReID

By considering ReID from an information retrieval perspective, there are usually three steps when applying deep metric learning. The first step is to derive a discerning embedding vector for representing a vehicle image. Then, to adopt a similarity metric, such as Euclidean or Cosine distances. And finally, to perform a search using the selected metric and a suitable acceleration data structure. Zhong et al. [18] emphasized the key role of re-ranking for improving accuracy given that most methods do not need additional training. They proposed k-reciprocal encoding to favor true matches during the search step. These authors carried out experimentation in well-known datasets for person ReID and showed that their encoding can improve retrieval performance by a fair margin. In fact, many works have adopted some re-ranking technique for improving their vehicle ReID results [4].

Liu et al. [15] [7] provided a baseline for VeRi-776 dataset by using FACT (Fusion of Attributes and Color feaTures) to combine a low-level color feature and high-level semantic attribute. These authors also tested several found in content-based image retrieval literature, such as BOW-SIFT, BOW-CN, AlexNet, AlexNet + BOW-CN, and GoogLeNet [19]. However, none of these methods outperformed FACT.

Hongye Liu et al. [6] proposed Deep Relative Distance Learning (DRDL) method to derive an end-to-end siamese CNN that learns both a 1024-dimensional feature vector representation and a similarity metric computation. DRDL is inspired by linear discriminant analysis and is built on top of VGGM-1024 [20] as *coupled clusters loss* to learn relative distances for vehicles. Similarly to triplet loss, this function considers distances to a cluster center, i.e., it can take more than three samples into account. It should be noted that their training procedure depends on vehicle model information, so DRDL cannot be applied directly to datasets missing such annotations.

PROVID (PROgressive Vehicle re-IDentification) [4] is a framework based on deep neural networks that uses the multimodal data, *i.e.*, license plates, camera locations, and contextual information, in large-scale video surveillance to perform vehicle re-identification. PROVID adopts re-ranking based on spatiotemporal information, which is measured in terms of a spatiotemporal similarity metric computed as the normalized product between ratios of temporal distance times the physical distance across two cameras considering the max-

imum differences. PROVID use 1024-dimensional embedding vectors extracted using GoogleLeNet [19] as representation.

Bai et al. [16] proposed group-sensitive-triplet embedding (GS-TRE), a deep metric learning approach that incorporates a group representation as an intermediate element between each vehicle and the samples. They proposed an online grouping strategy during training to build triplet samples at multiple granularities across different vehicles, i.e., samples of the same ID are partitioned into different groups, so the method could foster the learning of fine-grained features. They build on top of VGG-M-1024 [20], which contains 5 convolutional layers, 2 fully-connected layers, plus an L2 normalization layer that outputs a 1024-dimensional embedding vector.

Kumar et al. [10] investigated the use of contrastive and triplet loss functions for vehicle re-identification. These authors formalized batch sampling (BS) [1] and adopted MobileNet [21] as backbone while discarding the usual L2 normalization layer [17] when producing a 128-dimensional embedding vector. They carried out an extensive evaluation of its application to vehicle ReID to propose a strong baseline for this problem, which is competitive or outperforms recent methods.

As most re-identification methods rely on global appearance representations, He et al. [2] proposed a method for preserving part-regularized discerning features for enhancing the discrimination of very similar vehicles. They also built on top of ResNet50 [22], but introducing a detection branch for improving how local features are used for ReID. They also apply global average pooling and 1x1 convolution before producing a 256-dimensional feature embedding. The resulting framework was trained on an end-to-end basis to obtain remarkable ReID performance, thus surpassing most methods published so far.

Cosine-MGN (CMGN) [3] is based on Multi Granularity Network (MGN) [23]. CMGN is built on top of ResNet50 [22] and is one of the strongest ReID models found in literature. CMGN has two key modifications. First, the reduction block, or costume head, had its reduction block improved by replacing the pooling layer *MaxPool2d* with an *AdaptiveConcatPool2d*. A batch normalization layer was added before and after each 2D convolution as well and the weights were shared for each of 8 feature embedding vectors with 256 components, resulting in a 2048-dimensional representation. Second, an angular margin layer, called *Cosine* was used instead of all eight linear classification layers on MGN. The loss function of CMGN is a combination of Triplet Loss and Cosine CrossEntropy Loss with faster convergence.

D. Discussion

Vehicle Re-identification is an extremely active research field in which CNN and deep metric learning are fundamental tools. ResNet50 is a common building block in many promising results found in literature [2] [3]. Even though the MobileNet backbone is faster for real-time and mobile applications, it would be interesting to investigate how the approach in [10] can improve results when ResNet50 is adopted as the backbone network. VeRi sounds to be the most prominent dataset in literature for comparing methods since the seminal results in the field published in 2016. It should be noted that VeRi is distributed with a clear division of the samples between images for training and evaluation, providing a standard set for use as a query and test. Future works will likely turn their focus on even challenging datasets such as CityFlow-ReID and VeRi-Wild due to their resemblance to the unconstrained real-world problem.

As we can see from similar works found in literature, researchers usually resort to increasingly higher-dimensional feature embeddings trained with different strategies in order to improve their results [2] [16] [3] (see Table IV). It is worthy to mention that such improvement demands more processing power, especially when "plain" fully-connected layers are employed to compute the embeddings. On the other hand, there is also evidence in the literature that competitive methods can be obtained with relatively lower-dimensional embeddings and simpler backbone architectures [10]. Finally, a clever strategy for training the feature extraction, in particular how batches are constructed before these can be fed into the network, is also a key ingredient in most cases.

III. Our Approach

As stated beforehand, the main goal of our approach is to provide a viable solution for Vehicle ReID in the context of public safety in Brazil. Also, our project is carried out under hardware constraints, therefore the proposed re-identification system must be capable of operating in CPUs without hardware acceleration.

A. Overall System Architecture

Our system relies on cloud infrastructure and is designed to operate over an entire city with about 2,000 cameras at fixed locations in order to favor full-body shots from vehicles. Fig. 2 presents an overview of the proposed VeID system architecture. As one can see, captured images are sent to a `Frame Processing Service` along with a timestamp, camera ID, and position information. This service then resorts to a robust vehicle detection algorithm, such as SSD [13] or YOLO [12], that crops vehicle images. Naturally, the detection process has a crucial impact on the overall performance of the system, but this discussion is beyond the scope of this paper.

These image crops are then sent to a `Deep Metric Generation Service`, so each vector embedding is linked to the respective crop and metadata by a `Storage and Indexing Service`. There are other services available in the system, such as ALPR and vehicle color, model, and make classifiers, which can add more contextual, semantic information to the underlying representation. However, exploring such additional features goes beyond the scope of this paper.

An offline step is responsible for generating specific search datasets according to application criteria. For example, the system can produce an index for the "most wanted vehicles" or "target cars for an inspection". The index is built on top of

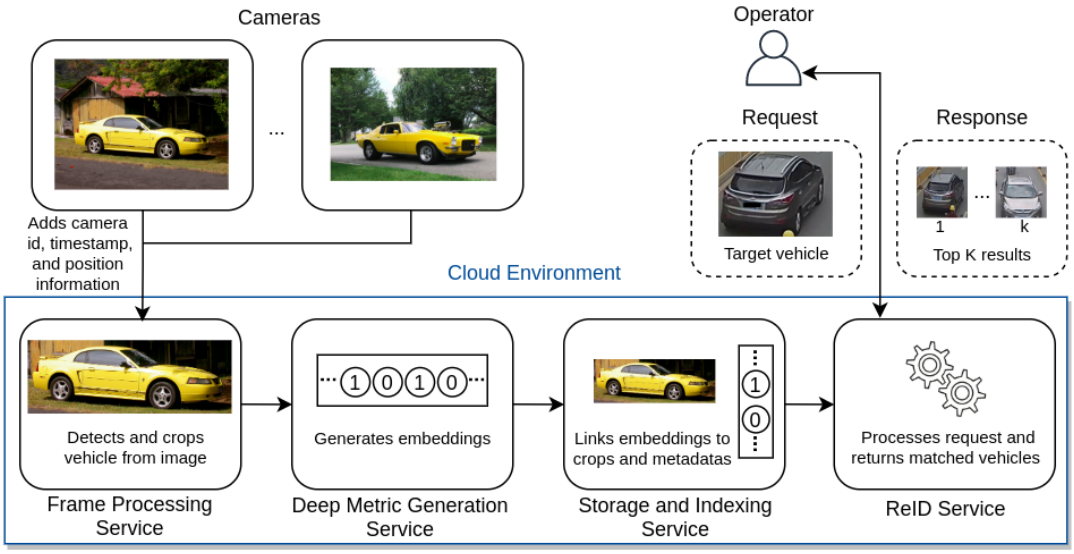

Fig. 2. Overview of our ReID system architecture.

in-memory ANN algorithms, which can also save the search structure to disk in order to publish that search index for use in remote devices or edge servers. A benefit from efficient ANN algorithms is that, once the index is built, the search can execute in real-time at a low cost given that such data fits target system memory and provides efficient search.

There is spatiotemporal and context information that can further improve results. However, this work focus on results obtained using solely feature embeddings and approximate nearest neighbors since this is the overall approach adopted by most comparable methods we could find in literature.

The built indexes are sent to the *ReID Service* to allow operators to perform vehicle searches through REST requests. The service receives a vehicle image as input and returns the Top_K most similar vehicle images.

B. Deep Metric Learning

We build on top of ResNet architecture based on the fact that it is an important element for both person and vehicle ReID. However, different from most works found in literature, we decided to adopt ResNet34 instead of Resnet50, as simpler networks achieved considerable performance for vehicle ReID [10]. Also, practical results[1] suggest that ResNet34 is capable of handling complex shapes such as faces even with further simplifications, something that could extend to vehicles. A final 128-dimensional embedding is extracted from the last layer fully-connected bottleneck.

$$\|f(x_1) - f(x_2)\| < d \tag{4}$$

$$\|f(x_a) - f(x_p)\|_2^2 < \|f(x_a) - f(x_n)\|_2^2 \tag{5}$$

Two loss layers were adopted for the sake of comparison: Coupled Loss (CL, Eq. 4), and Triplet Loss (TL, Eq, 5) [1], [17]. CL is defined in terms of a global distance threshold d, but also adding a margin ϵ for deciding whether two 128-dimensional embeddings, $f(x_1)$ and $f(x_2)$, of two vehicles, belong to the same id. Thus, if the distance between the embeddings is greater than d, then they belong to different vehicles.

TL is similar, but its purpose is ensuring that an embedding called *positive*, $f(x_p)$, lies closer to an *anchor*, $f(x_a)$, of the same vehicle id than an embedding of a different vehicle, $f(x_n)$, called *negative*. Two vehicles are different if the distance between their embeddings is high. Conversely, these are the same vehicle if that distance is low. High and low values are usually defined by a threshold.

TL typically requires an L2 normalization beforehand to avoid numerical issues during training [24], so the embedding vector is extracted from this additional layer. Training is carried out in two levels, as follows. At the training batch construction level, we perform a uniform random selection for obtaining N full-body images from V vehicles forming a batch tensor. The following preprocessing is applied for each color image: rotation from -4.5 to 4.5 degrees; scaling factor from 100 to 95%; displacement up to 1.5 the size at each dimension; resized to $150x150$ with no enhancements, different from [3]; and color distortion by combining random additive noise and gamma correction. Duplicate images are allowed, which favors both the matching of near-duplicate shots from the same vehicle and the challenges posed by difficult cases.

In fact, by using backpropagation during training, a network can only learn from cases in which the constraint imposed by a loss function is violated. Online sample mining similar to [25] is performed at the batch loss computation level, so the training procedure benefits from hard negatives, which typically improve quality and convergence. However, we also combine hard negatives with random samples, which, different

[1]http://vis-www.cs.umass.edu/lfw/results.html

from [10], also takes into account information from easy negatives.

C. Approximate Nearest Neighbors

The main idea behind ANN is to obtain a dramatic improvement of search performance, which comes at the cost of an acceptable loss in the accuracy, so an approximately correct but fast result is reported [26] [27]. This means that ANN methods can be used for different tasks, such as classification, tagging, and recommendation. In addition, there is clear evidence in the literature that, for some cases, ANN can also provide unexpectedly enhancement of results [28], [29]. We have chosen four representative state-of-the-art ANN methods that are widely used in both research and actual products [27], [30]–[32] to be evaluated in our strategy. Moreover, we also explore an exact approach to this problem [33].

FLANN [26] [30] was one of the first libraries to provide ANN methods. FLANN incorporates many improved nearest-neighbor search algorithms in metric spaces, such as the many variants of the kd-tree and locality sensitive hashing [26], [27]. FLANN adopts kd-tree, can scale to both high-dimensional features and very large datasets, and is quite efficient in building the indexing. Besides, this library features auto-tuning, although this process has observable limitations for handling high recall values [27].

ANNOY (Approximate Nearest Neighbors Oh Yeah) [27] was developed by Erik Bernhardsson and is used by Spotify for music recommendation. Its implementation resorts to an elegant Binary Space Partition (BSP) tree using hyperplanes, similar to those used for rendering purposes on early 3D graphics. A random forest of BSP trees is built to improve query results: the more trees, the more accurate results in less "performance (time) for higher accuracy (quality)". Default settings are to use $2 * D$, where D is dimensionality.

HNSW (Hierarchical Navigable Small World Graphs) [31] is a graph-based method that separates links between nodes in a search graph according to their length scale into different layers, thus performing searches in a multi-layer graph. Consequently, the evaluation of the nearest node occurs only on a needed fixed portion of the connections for each element. The main advantages of the HNSW method are its robustness and its support to continuous incremental indexing.

NGT (Neighborhood Graph and Tree for Indexing High-dimensional Data) is a graph-based ANN method initially proposed to work with large scale, high-dimensional vector space [32]. NGT improves k-nearest neighbor graph (KNNG) by adjusting path and edge degrees derived from a KNNG, i.e., an optimization in terms of the number of incoming edges and outgoing edges influencing the search accuracy and query time, respectively. In particular, the authors describe how a significant speedup can be obtained by removing edges that can be replaced by alternative search paths. NGT also supports adding and removing items to the indexing, which is an important feature for real-world applications. Experimental evaluation shows that NGT can compete or even outperform other methods [27], [32].

Finally, Blanco and Kumar [33] proposed *nanoflann*,for building KD-Trees of datasets describing point clouds and rotation groups. However, *nanoflann* does not focus on approximate nearest neighbor searches but the exact nearest neighbor performance by resorting to optimizations, such as adopting squared distances and customizable memory access policies.

IV. Experimental Evaluation

Our approach was implemented in C++ 11 on top of CUDA 10.0 using *Caffe*, then porting our training strategy to *dlib* after problems when producing the top-level batch generation. We also have implemented *ann-toolbox*, a wrapper for ANN algorithms relying heavily on templates to reduce performance footprint by accessing low-level parameters in detail, as opposed to ANN-Benchmarks written in Python [27]. Our results focus on VeRi and VehicleID: we consider these are well-suited for our specific application purposes since license plate information is omitted. Moreover, their overall structure is closer to the real-world data used by our system.

A. Experimental Setup

Our experiments were carried out in a PC equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor, 512GB SSD with nominal 396MB/s and 276MB/s R/W speeds, 2x8GB SODIMM DDR4 synchronous 2667 MHz RAM, NVIDIA TITAN Xp running on Ubuntu 18.04 LTS. Since some ANN methods do not resort to GPU acceleration, this feature was disabled for all algorithms in the ReID experimentation after the embedding vectors were computed for the sake of fair comparison.

B. Deep Metric Learning

Embedding models were trained over both VeRi and VehicleID. Our sample mining training strategy takes from 3 to 3.5 hours and 4 to 4.5 hours for CL (coupled loss) and TL (triplet loss) versions of our CNN, respectively, using $10x10$ NxV batches and standard SGD parameters for training until average loss cannot improve over 15,000 iterations. Triplet loss converges more slowly, requiring more training iterations ($\sim 166k$) than coupled loss ($\sim 148k$).

C. Comparing CL and TL Models

We compared the two best models considering vehicle matching and ReID tasks. The matching performance was computed exhaustively using data augmentation for tens of millions of comparisons between each vehicle id. We also used a model trained on VeRi for matching vehicles in VehicleID and vice-versa. ReID performance was evaluated with no data augmentation, despite better results that can be reported.

V. Results

Both models obtained promising matching accuracy on their respective training datasets. As the literature suggests, TL models consistently and significantly outperformed their CL counterparts, obtaining 99.34%, 95.74%, and 96.13% for VeRi

train, test, and query datasets against 96.25%, 92.66%, and 93.02% for CL models. Similar behavior is observed in the case of VehicleID (see Table I). Also, we could observe that TL models trained on a dataset (e.g., VeRi) were even better at generalizing their matching results to the other dataset (e.g., VehicleID) by a significant margin. Moreover, both CL and TL models trained on VeRi consistently obtained matching accuracy above 91% (see Table II).

TABLE I
VEHICLE MACTHING: COUPLED LOSS

Dataset	#classes	Train VeRi acc.	Train VehicleID acc.
VehicleID	800	91.56%	95.72%
VehicleID	1600	91.78%	95.74%
VehicleID	2400	91.79%	95.44%
VehicleID	3200	91.29%	95.67%
VehicleID	6000	91.86%	95.67%
VehicleID	13164	91.28%	95.58%
VeRi-Test	200	92.66%	73.38%
VeRi-Query	200	93.02%	74.89%

TABLE II
VEHICLE MACTHING: TRIPLET LOSS

Dataset	#classes	Train VeRi acc.	Train VehicleID acc.
VehicleID	800	93.98%	98.22%
VehicleID	1600	93.29%	97.59%
VehicleID	2400	94.26%	98.95%
VehicleID	3200	94.67%	98.79%
VehicleID	6000	93.53%	97.59%
VehicleID	13164	93.54%	97.75%
VeRi-Test	200	95.74%	88.63%
VeRi-Query	200	96.13%	88.79%

Experimental ReID performance results are summarized in Table III for both CL and TL over VeRi dataset. An extensive verification of parameters was carried out when using all ANN methods regarding retrieval quality, speed, and preprocessing to build the index. We emphasize that *nanoflann* actually computes *exact* nearest neighbors, hence its results serve as a baseline for comparison against brute-force approaches. HNSW is faster up to 2 orders of magnitude: average search took just $2ms$ for an entire query dataset, i.e., $1.1s$ for each query while NGT, ANNOY, FLANN, and *nanoflann* took $2.4\mu s$, $33.5\mu s$, $49.1\mu s$, and $92.3\mu s$, respectively. Considering the index build time, *nanoflann*, FLANN, HNSW, NGT, and ANNOY took $6\mu s$, $47.1\mu s$, $129.3\mu s$, $261.0\mu s$, and $684.1\mu s$ per item, respectively.

Surprisingly, CL embeddings outperform those obtained by triplet loss for ReID by a fair margin despite its lower vehicle pair matching accuracy, which we confirmed double-checked by disabling data augmentation in the matching task. Regarding mAP, all incarnations of TL-based methods are competitive with recent results found in literature (see Tables III and IV). Considering *HIT*, TL is still less effective than BS [10] unless NGT is used for ranked list retrieval: there is a substantial improvement from 88.98% to 98.09% concerning the $HIT@1$ metric. By adopting the optimized graph-based ANN search method, TL-NGT can present competitive performance. On the other hand, CL-NGT also benefits from this search strategy and outperforms all vehicle ReID methods listed in Table IV: this is an improvement of 1% and 2.33% over the state-of-the-art in terms of mAP and $HIT@1$. However, there are still challenging cases, even for human operators (see Fig. 3).

TABLE III
REID PERFORMANCE ON VERI

Method	**Loss**	**mAP**	**HIT@1**	**HIT@3**	**HIT@5**
ANNOY	CL	85.43	94.87	96.42	97.91
	TL	77.66	88.98	92.56	95.35
FLANN	CL	85.15	94.58	96.25	97.62
	TL	77.68	88.92	92.55	95.35
HNSW	CL	85.62	94.87	96.54	98.03
	TL	77.68	88.98	92.55	95.35
$nanoflann^*$	CL	85.60	94.87	96.54	98.03
	TL	77.68	88.98	92.55	95.35
NGT	CL	**86.20**	**98.93**	**99.17**	**99.52**
	TL	79.87	98.09	98.69	98.81

Fig. 3. Challenging situations found in both CL-NGT and TL-NGT. The query image is highlighted in blue, while the correct results are highlighted in green. The first two cases are difficult even for human operators: a careful examination is needed to understand why similar cars actually have different IDs. One can observe how hard is to find out that a little detail in the left rear window can evidence similarity between the correct results and the query image (*T2* and *C2*). The image aspect ratio is kept for better visualization of differences. Best viewed in color.

TABLE IV
COMPARISON OF RESULTS ON VERI-776.

Method	Backbone	ED	mAP	HIT@1	HIT@5
FACT [7]	-	-	18.75	52.21	72.88
PROVID [4]	GoogleLeNet	1024	27.77	61.44	78.78
BS [10]	MobileNet	128	67.55	90.23	96.42
GS-TRE [16]	VGGM1024	1024	59.47	96.24	98.97
PR [2]	ResNet50	256	74.30	94.30	98.70
CMGN [3]	ResNet50	2048	85.20	96.60	-
TL-NGT	ResNet34	128	79.87	98.09	98.81
CL-NGT	ResNet34	128	**86.20**	**98.93**	**99.52**

VI. CONCLUDING REMARKS

This paper proposes an effective approach for real-time vehicle ReID based on deep metric learning and ANN. Embedding models trained on VeRi are quite accurate for vehicle pair matching VehicleID. CL-NGT and TL-NGT show very promising results since these outperformed recent, state-of-the-art approaches in terms of mAP and *HIT* performance

metrics. Experimental results show that triplet loss actually performs better than coupled loss by a fair margin in two representative datasets. However, this apparent advantage did not hold for ReID task using ANN: mAP, $HIT@1$, and $HIT@5$ are consistently better for CL.

Our results suggest that the problems defined by triplet loss and re-identification are similar but not equivalent. A clear distance constraint found in coupled loss (Eq. 4) which is lacking in triplet loss also seems to favor ReID performance. So, researchers must be aware of counter-intuitive ReID performance issues when resorting to deep metric learning. Future works will focus on larger datasets [8], [9], other objects (bikes, animals, etc), reinforcement learning, model distillation, and the influence of object detection in ReID.

ACKNOWLEDGMENT

The authors would like to thank The Ceará State Foundation for the Support of Scientific and Technological Development (FUNCAP) for financial support (6945087/2019). The Titan Xp used in this research was donated by the NVIDIA Corporation. Portions of the research in this paper use the VehicleID dataset collected under the sponsor of the National Natural Science Foundation of China.

REFERENCES

[1] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[2] B. He, J. Li, Y. Zhao, and Y. Tian, "Part-regularized near-duplicate vehicle re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3997–4005, 2019.

[3] A. Ayala-Acevedo, A. Devgun, S. Zahir, and S. Askary, "Vehicle re-identification: Pushing the limits of re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 291–296, 2019.

[4] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 645–658, 2017.

[5] I. O. de Oliveira, R. Laroca, D. Menotti, K. V. O. Fonseca, and R. Minetto, "Vehicle re-identification: exploring feature fusion using multi-stream convolutional networks," *arXiv preprint*, vol. arXiv:1911.05541, pp. 1–11, 2019.

[6] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2167–2175, 2016.

[7] X. Liu, W. Liu, T. Mei, and H. Ma, "A deep learning-based approach to progressive vehicle re-identification for urban surveillance," in *European Conference on Computer Vision*, pp. 869–884, Springer, 2016.

[8] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8797–8806, 2019.

[9] Y. Lou, Y. Bai, J. Liu, S. Wang, and L. Duan, "Veri-wild: A large dataset and a new method for vehicle re-identification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3235–3243, 2019.

[10] R. Kumar, E. Weill, F. Aghdasi, and P. Sriram, "Vehicle re-identification: an efficient baseline using triplet embedding," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2019.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[14] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1224–1244, 2017.

[15] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2016.

[16] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L.-Y. Duan, "Group-sensitive triplet embedding for vehicle reidentification," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2385–2399, 2018.

[17] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[18] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1318–1327, 2017.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[20] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[23] G. Wang, Y. Yuan, X. Chen, J. Li, and X. Zhou, "Learning discriminative features with multiple granularities for person re-identification," in *Proceedings of the 26th ACM international conference on Multimedia*, pp. 274–282, 2018.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[25] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), vol. 2, pp. 119.1–119.11, BMVA Press, September 2016.

[26] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[27] M. Aumüller, E. Bernhardsson, and A. Faithfull, "Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," *Information Systems*, vol. 87, p. 101374, 2020.

[28] L. Boytsov, D. Novak, Y. Malkov, and E. Nyberg, "Off the beaten path: Let's replace term-based retrieval with k-nn search," in *Proceedings of the 25th ACM international on conference on information and knowledge management*, pp. 1099–1108, 2016.

[29] C. V. Gysel, M. De Rijke, and E. Kanoulas, "Neural vector spaces for unsupervised information retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 4, pp. 1–25, 2018.

[30] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[31] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[32] M. Iwasaki and D. Miyazaki, "Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data," *arXiv preprint arXiv:1810.07355*, 2018.

[33] J. L. Blanco and P. K. Rai, "nanoflann: a c++ header-only fork of flann, a library for nearest neighbor (nn) wih kd-trees," 2014.