Simulating Content Consistent Vehicle Datasets with Attribute Descent

Yue Yao\textsuperscript{1}  Liang Zheng\textsuperscript{1}  Xiaodong Yang\textsuperscript{2}  Milind Naphade\textsuperscript{2}  Tom Gedeon\textsuperscript{1}
\textsuperscript{1}Australian National University  \textsuperscript{2}NVIDIA

Abstract

We simulate data using a graphic engine to augment real-world datasets, with application to vehicle re-identification (re-ID). In order for data augmentation to be effective, the simulated data should be similar to the real data in key attributes like illumination and viewpoint. We introduce a large-scale synthetic dataset VehicleX. Created in Unity, it contains 1,209 vehicles of various models in 3D with fully editable attributes. We propose an attribute descent approach to let VehicleX approximate the attributes in real-world datasets. Specifically, we manipulate each attribute in VehicleX, aiming to minimize the discrepancy between VehicleX and real data in terms of the Fréchet Inception Distance (FID). This attribute descent algorithm allows content-level domain adaptation (DA), which has advantages over existing DA methods working on the pixel level or feature level. We mix adapted VehicleX data with three vehicle re-ID datasets individually, and observe consistent improvement when the proposed attribute descent is applied. With the augmented datasets, we report competitive accuracy compared with state-of-the-art results. The VehicleX engine is available at \url{https://github.com/yorkeyao/VehicleX}.

1. Introduction

Data simulation, as can be conveniently performed in graphic engines, provides valuable convenience and flexibility for computer vision research \cite{25, 27, 26, 34, 29}. One can simulate a large amount of data under various combinations of environmental factors even from a small number of 3D object/scene models. In order for simulated data to be effective in the real world, the domain gap should be addressed from two levels: content level and appearance level \cite{15}. While much existing work focuses on appearance (style)-level domain adaptation \cite{7, 12, 40}, we focus on the content level, i.e., learning to simulate data with similar content to the real data, as different computer vision tasks require varieties of image contents.

This paper investigates the content simulation of synthetic datasets, which can augment real-world datasets for the specific real-world task of vehicle re-ID. This task matches a query vehicle image with every database image so as to find the true matches containing the same vehicle. Here, based on the graphics software Unity, we propose an editable vehicle generation engine called VehicleX. Given a target real dataset, we propose a method allowing the VehicleX engine to generate a synthetic dataset with similar attributes to the real data, such as orientation, illumination, camera coordinates, and so on.

Our system is designed based on the following considerations. It is expensive to collect large-scale datasets for re-ID tasks. During annotation, one needs to associate an object across different cameras, a difficult and laborious process as objects might exhibit very different appearances in different cameras. In addition, there also has been an increasing concern over privacy and data security, which makes collection of large real datasets difficult. On the other hand, we can see that datasets can be very different in their content. Here content means the object layout, illumination, and background in the image. For example, the VehicleID dataset \cite{19} consists mostly of car rears and car fronts, while vehicle viewpoints in the VeRi-776 \cite{20} cover a very diverse range. This content-level domain gap might cause a model trained on VehicleID to have poor performance on VeRi. Most existing domain adaptation methods work on the pixel level or the feature level so as to allow the source and target domains to have similar appearance or feature distributions. However, these approaches are not capable of handling content differences, as can often be encountered when training on synthetic data and testing on real data.

The flexibility of 3D graphic engines allows us to 1) scale up the training data and 2) potentially reduce the content domain gap between simulated data and real-world data. In order to make effective use of our approach, we make contributions from two aspects. First, we introduce a large-scale synthetic dataset named VehicleX, which lays the foundation of our work. It contains 272 backbone models, and after coloring, creates 1,209 different vehicles. Similar to existing 3D synthetic datasets such as PersonX \cite{29} and ShapeNet \cite{4}, VehicleX has editable attributes and
Figure 1: System workflow. (Left:) given a list of attribute and their values, we use a renderer (i.e., Unity) for vehicle simulation. We compute the Frchet Inception Distance (FID) between the simulated and real vehicles to indicate their distribution difference. By updating the values of attributes using the proposed attribute descent algorithm, we can minimize FID along the training iterations. (Right:) we use the learned attributes values that minimize FID to generate simulated data to be used for re-ID model training.

is able to generate a large training set by varying object and environment attributes.

Based on the VehicleX platform, we propose an attribute descent method which automatically configures the vehicle attributes, such that the simulated data shares similar content distributions with the real data of interest. Specifically, we manipulate a range of five key attributes closely related to the vehicle. To measure the distribution discrepancy between the simulated and real data, we use the FID score and aim to minimize it. In each iteration, we optimize the values of attributes in a specific sequence, which usually terminates within two iterations of optimization. The system workflow is shown in Figure 1.

We show the efficacy of performing attribute descent on VehicleX by jointly training the resulting simulated data with real-world datasets. We show that the simulated training data with optimized attributes can consistently improve re-ID accuracy on the real-world test set. Under this augmentation scheme, we achieve very competitive re-ID accuracy compared with the state-of-the-art approaches, validating the effectiveness of learning from data synthesis.

2. Related Work

Vehicle re-identification has received increasing attention in the past few years, and many effective systems are proposed, such as those based on vehicle keypoints [16, 35], color [32] and viewpoints [41]. Moreover, useful loss functions are successfully adopted, such as the cross-entropy loss [38], the triplet loss [10] and the label smooth regularization (LSR) [31]. In this paper, our baseline system is built with commonly used loss functions with no bells and whistles. Depending on the camera conditions, location and environment, existing vehicle re-ID datasets usually have their own distinct characteristics. For example, images in the VehicleID [19] are either captured from the front or the back. In comparison, the VeRi-776 [20] includes a wide range of viewpoints. The recently introduced CityFlow [33] has distinct camera heights and backgrounds. Despite these characteristics, our proposed data simulation approach can effectively augment various re-ID datasets due to its strong ability in content adaptation.

**Appearance (style)-level domain adaptation.** Domain adaption is used to reduce the domain gaps between the distributions of two datasets. To our knowledge, the majority of work in this field focuses on discrepancies in image style, such as real vs. synthetic [2] and real vs. sketch [24]. For example, some use the generative adversarial network (GAN) to reduce the style gap between two domains [12, 28, 7]. Various constraints are exerted on the generative model such that useful properties are preserved during image translation. While these works have been shown to be effective in reducing the style domain gap, a fundamental problem remains to be solved, i.e., the content difference.

**Content-level domain adaption.** to our knowledge, has been discussed by a few existing works [15, 26]. This paper adopts their advantages and makes new contributions. On the one hand, we adopt the idea of Nataniel et al. [26] that represents attributes using predefined distributions. We are also motivated by Amlan et al. [15], who suggest that GAN evaluation metrics (e.g., KID [3]) are potentially useful to measure content differences. On the other hand, the two methods [15, 26] use gradient-based methods for attribute optimization, such as reinforcement learning and finite difference. The gradient descent based methods contain many random variables and are sensitive to learning rates.
To overcome the difficulty in training, we propose attribute descent which does not involve random variables and has easy-to-configured step sizes. Our optimization method is easy to train, and the objective function converges stably.

Learning from synthetic data. Due to low data acquisition costs, learning from synthetic data is an attractive way to increase training set scale. Many applications exist in areas such as semantic segmentation [12, 8], navigation [17], object re-identification [29, 32], etc. Usually, prior knowledge is utilized during data synthesis since we will inevitably need to determine the distribution of attributes in our defined environment. Richter et al. suggest that attribute randomness in a reasonable range is beneficial [34]. Even if it is random, we need to specify the range of random variables in advance. Our work investigates and learns these attribute distributions for vehicle re-ID tasks.

Automatic data augmentation. This paper is also related to automatic data augmentation [6, 9], which learns parameters for data augmentation techniques like random crop or image flipping. They share a similar objective with us in generating extra data for model training. The key difference is that they are not specifically designed for the problem of domain gaps, let alone on the content level.

3. VehicleX Generation Engine

We introduce a large-scale synthetic dataset generator named VehicleX that includes two components: (1) vehicles created using the graphics engine Unity and (2) a Python API that interacts with the Unity 3D engine.

VehicleX has a diverse range of vehicle models and identities, allowing it to be able to adapt to the variance of real-world datasets. It has 272 backbones that are hand-crafted by artists. The backbones include various vehicle types including sedan, minivan, police car, ambulance, fire truck, tank truck, etc. Each backbone represents a real-world model. From these backbones, we obtain 1,209 identities by adding various colors or accessories. A comparison of VehicleX with some existing vehicle re-ID datasets is presented in Table 1. VehicleX is three times larger than the synthetic PAMTRI dataset [32] and can potentially render an unlimited number of images and cameras.

In this work, we define a training mode and a testing mode of VehicleX. The training mode has a black background and is used for attribute descent learning (see Section 4); in comparison, the testing mode uses random images (e.g., from CityFlow [33]) as backgrounds, and generates attribute-edited images. In addition, to increase randomness, the testing mode contains street objects such as lamp posts, billboards and trash cans. Figure 2 shows the simulation platform, and some sample vehicle identities.

We build the Unity-Python interface using the Unity ML-Agents plugin [14]. It allows Python to modify the attributes of the environment and vehicles, and obtain the rendered images. With this API, users can easily obtain rendered images by editing attributes without needing expert knowledge about Unity. The code of this API will be released together with VehicleX.

4. Proposed Method

4.1. Attribute Distribution Modeling

Important attributes. For vehicle re-ID, we consider the following attributes to be potentially influential on the training set simulation and testing accuracy. Figure 3 shows examples of the attribute editing process.

- Vehicle orientation is the horizontal viewpoint of a vehicle and takes the value between 0° and 359°. In
the real world, this attribute is important because the camera position is usually fixed and vehicles usually move along predefined trajectories. Therefore, the vehicle orientations captured by a certain camera can be highly similar, and we would like to learn this attribute using VehicleX engine.

- **Light direction** is a competent modeling of the daylight as cars generally present in outdoor scenes. Here, we assume directional parallel light, and the light direction is modeled from east (0°) to west (180°), which is the moving trajectory of the sun. For simplicity, we set the light’s color as light yellow.

- **Light intensity** is usually considered a critical factor for re-ID tasks. We manually defined a reasonable range for intensity from bright to light.

- **Camera height** describes the vertical distance from the ground, and significantly influences viewpoints.

- **Camera distance** determines the horizontal distance from vehicles. This factor has a strong effect on the vehicle resolution since the resolution of the entire image is predefined as 1920×1080. Additionally, the distance has slight impacts on viewpoints.

Distribution modeling. We model the aforementioned attributes with Multivariate Gaussian or Gaussian Mixture Model (GMM). This modeling strategy is also used in Nataniel et al.’s work [26]. We denote the attribute list as: \( \mathcal{A} = (a_1, a_2, ..., a_N) \), where \( N \) is the number of attributes considered in the system, and \( a_i, i = 1, ..., N \) is the random variable representing the \( i \)th attribute.

For the vehicle orientation, we use a GMM with six components. This is based on our prior knowledge that the field-of-view of a camera covers either a road or an intersection. If we do not consider vehicle turning, there are rarely more than four major directions at a crossroad. For lighting conditions and camera coordinates, we use four independent Gaussian distributions, each modeling light direction, light intensity, camera height, and camera distance, respectively.

Therefore, given \( N \) attributes, we optimize \( M \) mean values of the Gaussians, where \( M \geq N \).

We speculate that the means of the Gaussian distributions or components are more important than the standard deviations, because means reflect how the majority of the vehicles look. Although our method has the ability to handle variances, this would significantly increase the search space. As such, we redefine the values of standard deviations using experience and only optimize the means of all the Gaussians \( \mu = (\mu_1, \mu_2, ..., \mu_M) \), where \( \mu_i \in \mathbb{R}, i = 1, ..., M \) is the mean of the \( m \)th Gaussian. As a result, given the means \( \mu \) of the Gaussians, we can sample an attribute list as \( \mathcal{A} \sim G(\mu) \), where G is a function that generates a set of attributes given means of Gaussian.

4.2. Optimization

The objective of our optimization step is to generate a dataset that has a similar content distribution with respect to a target real dataset.

Measuring distribution difference. We use the Frchet Inception Distance (FID) [11] to quantitatively measure the distribution difference between two datasets. The FID score was originally introduced to measure the performance of GANs by analyzing the difference between generated and real images. Adversarial loss is not used as the measurement since there exists a huge appearance difference between synthetic and real data, and the discriminator would easily tell the difference between real and fake.

Formally, we denote the sets of simulated data and real data as \( X_s \) and \( X_r \) respectively, where \( X_s = \{ R(A_1), \cdots, R(A_K) \mid A_k \sim G(\mu) \} \), and \( R \) is our rendering function through the 3D graphics engine working on a given attribute list \( \mathcal{A} \) that controls the environment. \( K \) is the number of images in the simulated dataset. For FID calculation, we employ the Inception-V3 network [31] to map an image into its feature space. We view the feature as a multivariate real-valued random variable and assume that it
follows a Gaussian distribution. To measure the distribution difference between two Gaussians, we use “

cam.” to denote camera height and distance. We initialize the attributes by setting orientation to right, the light intensity to dark, light direction to west, camera height to being equal to the vehicle, and camera distance to medium. The content of those images become more and more similar to (C) the target real images through the optimization procedure.

Formally, for our defined parameters \( \mu \) for attributes list \( \mathcal{A} \), the objective is to find,

\[
\mu = \arg \min FID(X_s, X_r),
\]

\[
X_s = \{ \mathcal{R}(A_1), \ldots, \mathcal{R}(A_K) \} | A_k \sim G(\mu^0_k),
\]

where \( \mu^0_k \) is the mean of the feature distribution of the simulated data, and \( \mu_r \) and \( \Sigma_r \) are from the real data.

\textbf{Attribute descent.} An important difficulty for attribute optimization is that the rendering function (through the 3D engine) is not differentiable, so the widely used gradient-descent based methods cannot be readily used. Under this situation, there exist several methods for gradient estimation, such as finite-difference [15] and reinforcement learning [26]. These methods are developed in scenarios where there are many attributes/parameters to optimize. In comparison, our system only contains a few parameters, allowing us to design a more stable and efficient approach that is sufficiently effective in finding a global minimum.

We are motivated by coordinate descent, an optimization algorithm that can work in derivative-free contexts [36]. The most commonly known algorithm that uses coordinate descent is \( k \)-means [22]. Coordinate descent successively minimizes along coordinate directions to find a minimum of a function. The algorithm selects a coordinate to perform the search at each iteration.

Using Eq. 1 as the objective function, we propose attribute descent to optimize each single attribute in the attribute list. Specifically, we view each attribute as a coordinate in the coordinate descent algorithm. In each iteration, we successively change the value of an attribute to search for the minimum value of the objective function.

\textbf{Discussions.} Our method can be seen as an analogy to \( k \)-means. In \( k \)-means, the search for the minimum of the objective function is done by iterating between the E-step and M-step. The E-step or M-step can be viewed as searching along a coordinate. In our method, we view each Gaussian mean (associated with attributes) as a coordinate and search for the minimum along it.
Existing gradient based methods [15, 26] involve many intermediate random variables to be optimized and are relatively sensitive to learning rates. Moreover, FID measures the distribution difference of two datasets, so the non-differentiable supervision signal provided by FID is far less precise than cross-entropy loss. These difficulties potentially over-complicates training in a simple environment like ours. Our method does not contain any random variables, and the step size for optimizing each attribute is stable. Therefore, attribute descent converges stably and much easier to train. But we speculate that when simulating more complex environments where the search space is much larger, reinforcement learning or finite-difference could also be effective.

5. Experiment

5.1. Datasets and Evaluation Protocol

Datasets. We use three real-world datasets for evaluation. The VeRi-776 dataset [20] contains 49,357 images of 776 vehicles captured by 20 cameras. The vehicle viewpoints and illumination cover a diverse range. The training set has 37,778 images, corresponding to 576 identities; the test set has 11,579 images of 200 identities. There are 1,678 query images. The train/test sets share the same 20 cameras. VehicleID [19] is larger scale, containing 221,567 images of 26,328 identities. Half of the identities are used for training, and the other half for testing. Officially there 6 test splits for the gallery. When performing the re-ID task, one random image from each identity will be used to form the gallery and the rest will become the query. Such a procedure will be repeated ten times and numbers reported are averaged across these ten times. Here, we use VehicleID (large) which has 2,400 image as the gallery. CityFlow [33] has more complex environments, and it has 40 cameras in a diverse environment where 34 are used for the training set. This dataset has in total 666 IDs where half are used for training and the rest for testing. In order to perform attribute learning on the CityFlow test set, we manually labeled the camera information on the CityFlow test set.

Evaluation metrics. We use the mean average precision (mAP) and rank-1 accuracy to measure the vehicle re-ID performance. The average precision records the area under the precision-recall curve, and the rank-1 accuracy denotes the success rate of the top-1 match.

5.2. Implementation Details

Data generation. Vehicle re-ID datasets like VeRi-776 and CityFlow are naturally divided according to camera views. Because a specific camera view usually has stable attribute features (e.g., viewpoint), we perform the proposed attribute descent algorithm on each individual camera, so as to simulate images with similar content to images from each camera. For CityFlow, because the training set and test set contain entirely different cameras, we do the optimization on the test set. Note here that we do not use any ID labels from the test set, just the images themselves. For example, we optimize 20 attribute lists for the VeRi-776 test set, which has 20 cameras. An exception is for the VehicleID dataset. Its training set and test set have similar front and rear views. As such, we only optimize a single attribute list.

Moreover, during data simulation, we follow the same ratio of image number among different cameras in the target datasets. To explain, in the VeRi test set, the ratio of image number between camera 4 and camera 8 is approximately 1 : 1. So we also simulate a similar number of images with attributes learned from cameras 4 and 8. For CityFlow, we manually label the cameras in the test set because they are not provided. Note that camera labels can be conveniently obtained in practice.

Image style transformation. We apply the SPGAN [7] for image style transformation, which is a state-of-the-art algorithm in unsupervised domain adaptation for person re-ID. Results are shown in Figure 5. In our implementation, each image is sized to $256 \times 256$. We pretrained all image translation models for three datasets using random attribute, with 112,042 images as source domain and the training set in three vehicle datasets as target domain separately. For performing learned attributes domain adaption, we directly inference the learned attributes images, based on the fact that our learned attributes are a subset of the random range.

Baseline configuration. For VeRi and VehicleID, we use ID-discriminative embedding (IDE) [38]. We also adopt the part-based convolution baseline (PCB) [30] for VeRi for improved accuracy. In PCB, we horizontally divide the pic-
Table 3: Method comparison on VehicleID. Our method is built on IDE [38] with the cross-entropy (CE) loss. Our method (learned attributes) consistently improves over both the baseline and random attributes, and is very competitive compared with the state of the art. “R” means training use real data only. “R+S” denotes that both simulated data and real data are used in training.

Table 4: Method comparison on the VeRi-776 dataset. In addition to some state-of-the-art methods, we summarize results of our method on top of two baselines, i.e., IDE [38] and PCB [30].

Table 5: Method comparison on CityFlow. Our baseline is built with a combination of the cross-entropy loss and the triplet loss. We report the rank-1, rank-20 and the mAP (rank-100) calculated by the online server.

5.3. Simulated Data for Data Augmentation

Parameter analysis. An important hyperparameter under the data augmentation setting is the ratio between the numbers of simulated data and real data. We vary this ratio from \((0, 0.25, 0.5, 0.75, 1)\) and report re-ID accuracy on VeRi in Figure 6. First, we find that under an appropriate ratio (e.g., between 0.25 and 1), the data augmentation strategy is consistently higher than training with real data only. This supports our idea of simulating images to augment the real data. Second, we observe that setting the ratio to 0.75 achieves the best result on VeRi. In our experiment, we empirically use 0.65 for CityFlow and 0.4 for VehicleID.

Simulated data with randomly distributed attributes does not work consistently well. Several existing works indicate that random attributes are beneficial for data simulation in real-world tasks [34, 32]. We evaluated on the effectiveness of this strategy in Table 4. On VeRi-776, the use of random attributes slightly improves over the IDE and PCB baselines. We obtain mAP improvements of +0.21% and +0.67% under the IDE baseline and PCB baseline, respectively. For CityFlow results in Table 5, we see a significant mAP improvement of +1.82% under the IDE baseline.

Interestingly, random attributes barely improve the performance on VehicleID. For example, when the medium test set is used, random attributes lead to a drop of 0.26% in mAP. In fact, while VeRi-776 and CityFlow have diverse attribute distributions (e.g., nearly all the orientations are included), VehicleID mostly contains rear and front orien-
Figure 6: Evaluation of the ratio between the number of simulated images and real images in the training set. We report mAP and rank-1 accuracy on VeRi obtained by learned attributes. The dashed green line presents mAP of the IDE baseline trained with only the real data. When the ratio is between 0.25 and 1, data augmentation benefits the baseline accuracy.

Table 6: Comparison of mAP and rank-1 accuracy on VeRi when training with only simulated data. The IDE baseline is trained with only the real data. We report the results on the VeRi-776 in Table 6 and Table 7. Three observations are made.

First, if we compare the numbers in Table 4 and Table 5, we clearly observe that style DA brings consistent improvement and thus is necessary.

Second, the benefit of attribute learning is more significant than joint training. Because the training set consists of only the simulated data, the higher quality of attributes will have a more direct impact on the re-ID results. Under the settings of Section 5.3 and Section 5.4, the improvements of learned attributes over random attributes are +1.49% and +3.87% in rank-1 accuracy, respectively.

Last, from Table 7, when the number of simulated data increases, we can observe accuracy improvement, too. This suggests the potential of simulated data for future research. Since training with only the simulated is not our major focus, we do not explore further along this direction here.

5.4. Training with Simulated Data Only

Training with simulated data only. In the experiment above, our simulated data is mixed with the real data. Here, we evaluate re-ID accuracy when only simulated data is used for training. We report the result on the VeRi-776 in Table 6 and Table 7. Three observations are made.

First, if we compare the numbers in Table 4 and Table 6, it is obvious that training with only the simulated data has much lower accuracy. For example, under random attributes, data augmentation is higher than the simulation-only scheme by a significant margin of +44.97% in rank-1 accuracy. As such, the inclusion of real data is essential for achieving competitive accuracy. However, as graphic engines become stronger, it could be possible for simulation data alone to contribute to real-world systems.

Second, the benefit of attribute learning is more significant than joint training. Because the training set consists of only the simulated data, the higher quality of attributes will have a more direct impact on the re-ID results. Under the settings of Section 5.3 and Section 5.4, the improvements of learned attributes over random attributes are +1.49% and +3.87% in rank-1 accuracy, respectively.

6. Conclusion

This paper addresses the domain gap problem from the content level. That is, we automatically edit the source domain image content in a graphic engine so as to reduce the...
content gap between the simulated images and the real images. We use this idea to study the vehicle re-ID task, where the usage of vehicle bounding boxes decreases the set of attributes to be optimized. Fewer attributes-of-interest allow us to optimize them one by one using our proposed attribute descent approach. We show that the learned attributes bring about improvement in re-ID accuracy with statistical significance. Moreover, our experiment reveals some important insights regarding the usage of simulated data, e.g., style domain adaptation is beneficial.

We will continue investigating data simulation and its role in computer vision. Potential directions include more effective objective functions, alternative derivative-free optimization methods, and new recognition training schemes.

7. Appendix

In this appendix, we first detail the attribute descent algorithm. Then, we demonstrate the effectiveness of successively optimizing each attribute and using Fréchet Inception Distance (FID) as the measure of distribution difference. At last, we provide the statistical significance of the results on three datasets.

7.1. Implementation Details

In this section, we provide a detailed algorithm and the hyperparameter setting for the proposed attribute descent method. First, we present a step-by-step instruction for the proposed attribute descent method in Algorithm 1, where we successively optimize each attribute by minimizing the FID value. Second, we set the hyperparameters as follows.

### Table 6: Method comparison on VeRi-776 when only using simulated data for training. The label smooth regularization (LSR) [31] is used. SPGAN is used [7] for style domain adaptation for both our synthetic dataset and VehicleID. “R” means training with real data only, and “S” represents training using simulated data only. VID→VeRi shows the result trained on VehicleID, test on VeRi.

| Data    | Method       | Rank-1 | Rank-5 | mAP  |
|---------|--------------|--------|--------|------|
| R       | ImageNet     | 30.57  | 47.85  | 8.19 |
| R       | VID→VeRi     | 59.24  | 71.16  | 20.32|
| S       | Random Attr. | 44.40  |        |      |
| S       | Learned Attr.| 16.99  |        |      |
| S       | - Style DA   |        |        |      |

Table 7: The impact of the number of training images when only the simulated data is used for training. We use VeRi for evaluation.

| Data    | #Train Img | Rank-1 | Rank-5 | mAP  |
|---------|------------|--------|--------|------|
| S       | 10,000     | 43.38  | 61.32  | 16.99|
| S       | 20,000     | 44.40  | 62.57  | 18.55|
| S       | 30,000     | 48.33  | 66.33  | 19.63|

The synthetic dataset size $K$ is set to 400. That is, we generate 400 images each iteration. For attribute descent iteration, we use $J = 2$ for the VehicleID [19] dataset and $J = 1$

**Algorithm 1** Attribute Descent Algorithm

**Input:** Parameter vector $\mu^0 = (\mu_1^0, \cdots, \mu_M^0)$ for attribute distribution function $G$, search space $S = \{S_1, \cdots, S_M\}$ for each element in $\mu = (\mu_1, \cdots, \mu_M)$, render function $R$ and target dataset $X_r$.

**Hyperparameters:** $J$ iterations for attribute descent and synthetic dataset size $K$.

**Begin:**

OptFID = $\infty$; \hspace{1cm} // Initialize the optimal FID value

for $j = 1$ to $J$ do

for $i = 1$ to $M$ do

for $z \in S_i$ do

Traverse search space $X_s = \{R(A_1), \cdots, R(A_K)\} \mid A_k \sim G(\mu_1, \cdots, \mu_i, z, \mu_{i+1}, \cdots, \mu_M)$

Generate synthetic dataset $X_s = \{R(A_1), \cdots, R(A_K)\} \mid A_k \sim G(\mu_1, \cdots, \mu_i, z, \mu_{i+1}, \cdots, \mu_M)$

Score = FID($R(X_s, X_r)$) \hspace{1cm} // FID calculation

if Score < OptFID then

OptFID = Score \hspace{1cm} // Update FID

$\mu_i^j = z$ \hspace{1cm} // Update parameters

end if

end for

end for

end for
for the VeRi [20] and the CityFlow [33] datasets. For FID calculation only, all images are resized into $64 \times 64$ since it is nearly the minimal size of the real-world dataset. FID is calculated by the 2048-dimensional feature vector from the Inception-V3 [31] network. The initial parameter vector $\mu_0$ of attribute generation function $G$ is set as the minimal value in the search space (see Figure 4 of original paper).

### 7.2. Effectiveness of Attributes Descent

In this section, we prove the effectiveness of the proposed attribute descent method. As shown in Algorithm 1, we successively optimize each attribute by minimizing the FID value. Along the process of optimizing each attribute, the FID value decreases (lower is better) and the re-ID accuracy (mAP) increases (Figure 8). Based on this, we have some observations.

- FID can measure the distribution difference, and thus can be used as an objective function to optimize the attributes of synthetic data;
- The inclusion of synthetic data can improve the re-ID accuracy, even with initialize attributes parameter vector $\mu^0$ for data generation;
- Our attribute descent algorithm can successively find a more suitable value for each attribute. Synthetic data generated from these attributes is more beneficial to re-ID feature training, resulting in higher mAP.

### 7.3. Statistical Significance Analysis

Learned attributes bring statistical significant improvements on re-ID accuracy (mAP) (Figure 9). Specifically, improvements on the CityFlow, VehicleID large and VehicleID small datasets are statistically very significant (i.e., $0.001 < p$-value $< 0.01$).

### References

[1] Yan Bai, Yihang Lou, Feng Gao, Shiqi Wang, Yuwei Wu, and Ling-Yu Duan. Group-sensitive triplet embedding for vehicle reidentification. *IEEE Transactions on Multimedia*, 20(9):2385–2399, 2018. 7

[2] Slawomir Bak, Peter Carr, and Jean-Francois Lalonde. Domain adaptation through synthesis for unsupervised person re-identification. In *ECCV*, 2018. 2

[3] Mikolaj Binkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *ICLR*, 2018. 2

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[5] Ruihang Chu, Yifan Sun, Yadong Li, Zheng Liu, Chi Zhang, and Yichen Wei. Vehicle re-identification with viewpoint-aware metric learning. In *ICCV*, 2019. 7, 8

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 3

[7] Weijian Deng, Liang Zheng, Qixiang Ye, Guoliang Kang, Yi Yang, and Jianbin Jiao. Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification. In *CVPR*, 2018. 1, 2, 6, 8, 9

[8] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 3

[9] Mingyang Geng, Kele Xu, Bo Ding, Huaimin Wang, and Lei Zhang. Learning data augmentation policies using augmented random search. *arXiv preprint arXiv:1811.04768*, 2018. 3

[10] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 2

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 4

[12] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 1, 2, 3

[13] Tsung-Wei Huang, Jiakai Cai, Hao Yang, Hung-Min Hsu, and Jenq-Neng Hung. Multi-view vehicle re-identification using temporal attention model and metadata re-ranking. In *CVPR Workshop*, 2019. 8

[14] Arthur Juliari, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. 3
[15] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusinak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. 2019. 1, 2, 5, 6

[16] Pirzah Khorraramshahi, Amit Kumar, Neeraj Peri, Sai Saketh Rambhatla, Jun-Cheng Chen, and Rama Chellappa. A dual path model with adaptive attention for vehicle re-identification. 2019. 2, 7

[17] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual AI. arXiv preprint arXiv:1712.05474, 2017. 3

[18] Ratnesh Kumar, Edwin Weill, Farzin Aghdasi, and Parthasarathy Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. arXiv preprint arXiv:1901.01015, 2019. 7

[19] Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In CVPR, 2016. 1, 2, 3, 5, 6, 9

[20] Xinchen Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. Large-scale vehicle re-identification in urban surveillance videos. In ICME, 2016. 1, 2, 3, 6, 10

[21] Xiaobin Liu, Shiliang Zhang, Qingming Huang, and Wen Gao. Ram: A region-aware deep model for vehicle re-identification. In ICME, 2018. 7

[22] Stuart Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137, 1982. 5

[23] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In CVPR Workshops, 2019. 7

[24] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In ICCV, 2019. 2

[25] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In ECCV, 2016. 1

[26] Nataniel Ruiz, Samuel Schulter, and Mannmohan Chandraker. Learning to simulate. In ICLR, 2019. 1, 2, 4, 5, 6

[27] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. International Journal of Computer Vision, pages 1–20. 1

[28] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In CVPR, 2017. 2

[29] Xiaoxiao Sun and Liang Zheng. Dissecting person re-identification from the viewpoint of viewpoint. In CVPR, 2019. 1, 3

[30] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In ECCV, 2018. 6, 7

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016. 2, 4, 9, 10

[32] Zheng Tang, Milind Naphade, Stan Birchfield, Jonathan Tremblay, William Hodge, Ratnesh Kumar, Shuo Wang, and Xiaodong Yang. Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data. In ICCV, 2019. 2, 3, 7

[33] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In CVPR, 2019. 2, 3, 6, 10

[34] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Janmani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In CVPR Workshops, 2018. 1, 3, 7

[35] Zhongdao Wang, Luming Tang, Xihui Liu, Zhuliang Yao, Shuai Yi, Jing Shao, Junjie Yan, Shengjin Wang, Hongsheng Li, and Xiaogang Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In ICCV, 2017. 2, 8

[36] Stephen J Wright. Coordinate descent algorithms. Mathematical Programming, 151(1):3–34, 2015. 5

[37] Linjie Yang, Bing Luo, Chen Change Loy, and Xiaoxu Tang. A large-scale car dataset for fine-grained categorization and verification. In CVPR, 2015. 3

[38] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars: A video benchmark for large-scale person re-identification. In ECCV, 2016. 2, 6, 7

[39] Zhedong Zheng, Tao Ruan, Yunchao Wei, and Yi Yang. VehicleNet: Learning robust feature representation for vehicle re-identification. In CVPR Workshops, 2019. 7

[40] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, and Yi Yang. Camera style adaptation for person re-identification. In CVPR, 2018. 1

[41] Yi Zhou and Ling Shao. Aware attentive multi-view inference for vehicle re-identification. In CVPR, 2018. 2, 8