Attribute Descent: Simulating Object-Centric Datasets on the Content Level and Beyond

Yue Yao, Member, IEEE, Liang Zheng, Senior Member, IEEE, Xiaodong Yang, Milind Napthade, and Tom Gedeon, Senior Member, IEEE

Abstract—This article aims to use graphic engines to simulate a large number of training data that have free annotations and possibly strongly resemble to real-world data. Between synthetic and real, a two-level domain gap exists, involving content level and appearance level. While the latter is concerned with appearance style, the former problem arises from a different mechanism, i.e., content mismatch in attributes such as camera viewpoint, object placement and lighting conditions. In contrast to the widely-studied appearance-level gap, the content-level discrepancy has not been broadly studied. To address the content-level misalignment, we propose an attribute descent approach that automatically optimizes engine attributes to enable synthetic data to approximate real-world data. We verify our method on object-centric tasks, wherein an object takes up a major portion of an image. In these tasks, the search space is relatively small, and the optimization of each attribute yields sufficiently obvious supervision signals. We collect a new synthetic asset VehicleX, and reformat and reuse existing the synthetic assets ObjectX and PersonX. Extensive experiments on image classification and object re-identification confirm that adapted synthetic data can be effectively used in three scenarios: training with synthetic data only, training data augmentation and numerically understanding dataset content.

Index Terms—Attribute descent, data simulation, object-centric datasets.

I. INTRODUCTION

DATA synthesis that can be conveniently performed in graphic engines provides invaluable convenience and flexibility for the computer vision community [1], [2], [3], [4], [5]. A large number of training data can be synthesized under different combinations of environmental factors from a small number of 3D object models. Despite this convenience, a large domain gap exists between synthetic data and real-world data, which can substantially decrease accuracy when a model trained on synthetic data is tested with real data [3], [6]. The domain gap problem can be addressed from the appearance level [6] or the content level (Fig. 1). The former often translates image style by using a neural network [7], whereas the latter modifies the image content via manipulating editable attributes directly in graphic engines.

We are motivated by the following considerations. First, collecting large-scale real-world datasets with manual labels is expensive. For example, in a multi-camera system like object re-identification (re-ID), an object must be associated across multiple different cameras to obtain ground truth labels: this process is extremely difficult and laborious because objects usually vary in appearance under different cameras. Privacy and data security concerns further add overheads to this process.

Second, the domain gap between datasets exists not only on the appearance level [7], but also on the content level [6]. Take vehicle re-ID as an example. Images in the VehicleID dataset [8] mainly exhibit car rears and fronts, whereas the VeRi-776 dataset [9] covers more diverse viewpoints. This difference in viewpoints is considered an example of content-level

Fig. 1. Domain adaptation on the content level (a) and appearance level (b). With real data as a target, content-level domain adaptation optimizes image content attributes such as camera viewpoint and illumination. In comparison, appearance-level and feature-level domain adaptation typically use image style transfer (shown) or feature alignment (not shown). Specifically, given a list of attributes and their values, we use the renderer Unity for training data synthesis. Afterward, we compute the FID between the synthetic and real images to reflect their distribution difference. By updating the values of attributes using the proposed attribute descent algorithm, we can minimize the FID along the training iterations. We use the optimized attributes to generate synthetic data, which can be used to replace or scale up real-world training data.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
discrepancy between datasets. As a result, models trained on VehicleID [8] have a significant accuracy drop when tested on VeRi-776 [9]. Existing domain adaptation methods on the appearance level can alleviate this problem and improve accuracy, but are essentially incapable of handling the content differences.

Given these considerations, this article proposes an attribute descent algorithm to generate synthetic training data 1) in a large scale and inexpensive manner, 2) with a reduced content domain gap with real-world data. In a nutshell, as shown in Fig. 1(a), the proposed attribute descent algorithm can automatically configure simulator attributes, such as camera viewpoint, lighting direction, and object placement, so that the synthesized data are close to real-world data as measured by the Fréchet Inception Distance (FID) [10]. These optimized attributes are subsequently used for training set generation or augmentation, where style-level image translation can be leveraged to further reduce domain discrepancy. To make the above process happen, we define a controllable simulation environment in the graphic engine with editable object and environment attributes, allowing us to generate large training sets by varying these attributes.

We evaluate the effectiveness of synthetic data generated by attribute descent on object-centric tasks: image classification, person re-ID and vehicle re-ID. In these tasks, an image usually contains a single object of interest which is placed approximately in the image center (through either automatic detection or manual cropping). Specifically, image classification aims to distinguish different classes for an input object of interest, while object re-ID targets at differentiating object (person, vehicle) identities. For each of the tasks, we build a synthetic object-centric filming scenario, which contains an object of interest, a camera and a direct lighting source. Compared with complex-scene applications such as semantic segmentation, object-centric tasks require fewer 3D object assets and have simpler layout relationships between assets, smaller environments and fewer attributes to be controlled. These favorable properties simplify the problem addressed and enable in-depth method analysis. More discussions of the application scope of our method are provided in Section VI.

We identify three scenarios in which data synthesized by attribute descent can be effectively applied: training with adapted synthetic data only, augmenting real training data with synthetic data, and visualizing content-level dataset bias. First, after optimizing synthetic data to approximate target distribution (real world), we directly use synthetic data to train object-centric task models for recognition or re-ID. Second, we add the optimized synthetic data to real-world data to enlarge the training set, which is used for training the task model and improving accuracy. In both applications, we demonstrate that optimized synthetic data are superior to those randomly generated and that the inclusion of the former consistently enhances real-world training data. Furthermore, the two applications suggest the existence of content bias in object-centric datasets, and a third interesting application is understanding dataset content by using obtained attribute value distributions. For example, we visualize that vehicle orientations on a normal two-way road are usually bi-modal, exhibiting two major angles, whereas those in an intersection are more diverse.

Experimentally we show that attribute descent is superior to using random attributes for training set synthesis, when the training is either with synthetic data only or has both synthetic and real data. Moreover, we compare attribute descent with existing gradient-free optimization techniques, namely Bayesian optimization, reinforcement learning, evolutionary algorithm, and random search. We observe that attribute descent leads to a consistently lower domain gap between the generated training set and target set, consequently yielding the model higher accuracy than the competing methods. Additionally, we report that attribute descent promotes more stable convergence than the competing methods. The above indicates the effectiveness of attribute descent for syn2real content-level domain adaptation.

This article is an extension of our conference publication [11]. Three major differences are presented. First, while only vehicle re-ID is studied in [11], our current system is capable of additionally improving person re-ID and image classification tasks. For these new tasks, we collect/extend new/existing 3D assets in the person and generic object domains and customize them into our pipeline. Second, comprehensive geometric modeling of object-camera placement. This introduces an additional in-plane rotation attribute that allows us to model object orientation more flexibly, particularly for generic objects. Third, by approximating real-world datasets with synthetic counterparts, we show that certain aspects of dataset content in real-world data can be numerically understood and visualized by this synthetic proxy. Interesting experimental observations are presented and discussed.

II. RELATED WORK

Realistic appearance generation and domain adaptation: Domain adaptation is often achieved by reducing the domain gaps between distributions. To date, a majority of works in this field has focused on discrepancies in appearance or feature level. For the former, a considerable body of research use the cycle generative adversarial network (CycleGAN) [12] and its variants to reduce the appearance gap [7], [13], [14], [15], [16]. The latter models the dependence between two domains by means of feature-level statistics [17], and many moment matching schemes are studied to learn a shared feature representation [18], [19], [20], [21], [22], [23]. Although these works are shown to be effective in reducing the appearance or feature domain gap, a fundamental problem remains to be solved, i.e., the content difference.

Learning from simulated 3D data: Data simulation is an inexpensive way of increasing a training set scale while providing accurate image labels, flexibility in content generation and high resolution. Learning from simulated data finds it applications in image recognition [24], re-identification [5], [27], semantic segmentation [7], [28], [29], navigation [30] and detection [6], [31]. Existing knowledge or estimation of data distribution is usually required during data synthesis in the graphic engine. Some applications directly take what is presented in existing video games such as GTA5, which have pre-defined scenes and objects [1], [24], [32], [33], [34], [35]. Others manually create their own simulation environments and objects [28], [30].
and find it is beneficial to use random attribute within a reasonable range to create random content [4], [39], [40]. However, despite being called “random”, the range of random variables still must be specified manually according to experience. Instead, we aim to learn attribute distributions more automatically or with less human experience.

Automatic 3D content creation: Many works aim to automatically create realistic 3D models [41], [42], focusing on intrinsic properties of 3D models such as backbones, geometry and surface. The studied object models include faces [43], persons [44], [45], vehicles [42] and furniture [41]. Departing from these works that mainly study object synthesis, we aim to use the optimized 3D models for downstream task training. Content creation for task model training. Several recent studies have attempted to automatically generate 3D content for training task models [3], [6], [29], [46]. They are closest to our work. For example, Kar et al. and Devaranjan et al. simulate traffic for training vehicle detection networks [6], [46]. Xue et al. and Ruiz et al. construct street scenes for semantic segmentation model training [3], [29]. Many of these works use reinforcement learning (RL) based algorithms to optimize attributes [3], [6], [29], [46]. In our article, we find RL less effective in object-centric tasks, in which a relatively small number of attributes needed to be optimized. In this sense, our method is an alternative to existing ones and particularly effective in object-centric tasks (see comparisons in Table VIII).

In object-centric vision tasks, a single object of interest usually appears in an image, because manual cropping or detection is used to place the object approximately in the image center. Specifically, the basic image classification problem aims to distinguish different classes for an input image [47], [48], [49], [50]. Another task that we consider is object re-ID [9], [51], [52], which has many robust systems proposed recently [27], [53], [54], [55]. When experimenting on these object-centric tasks, we adopt existing architectures and loss functions with no bells and whistles.

Dataset bias is a critical reason for compromised model performance [56]. For example, many classification datasets [47], [48], [57] are collected from public user repositories, such as Flickr, which may have content bias in terms of object placement, background, rotation, occlusion, lighting conditions etc. The bias may explain why models exhibit lower accuracy on test sets that have a different content bias from the training set [58]. There are some existing approaches for dataset bias visualization. For example, t-distributed stochastic neighbor embedding (t-SNE) can exhibit feature-level data distributions [59]. In comparison, we aim to visualize bias through attributes, i.e., obtaining attribute distributions numerically and drawing them in graphs.

Similarly, depending on the camera condition, location and environment, existing object re-ID datasets usually have their own distinct characteristics or bias [5]. For example, vehicle images in the VehicleID dataset [8] are either captured from car front or back, whereas the VeRi dataset [9] includes a much wider range of viewpoints. Beyond dataset-dataset differences, large differences also exist between cameras in a single dataset [60]. For example, a camera filming a cross road naturally has more vehicles orientation than a camera on a straight road. In this article, we leverage such characteristics to learn attributes for each camera and simultaneously visualize the attribute distributions.

III. SIMULATION ENVIRONMENT

Overall, we aim to build a simulation environment with editable attributes for building large-scale synthetic datasets that are close to the real world. To achieve this, we collect a large number of 3D objects (Section III-A), build a camera model in the graphic engine, define a set of editable attributes (Section III-B), and acquire synthetic images by varying these attributes (Section III-D).

A. 3D Asset Acquisition

For object-centric tasks, we use 3D assets from three sources. Their details are provided below as well as in Table I, and sample assets are shown in Fig. 2.

We reformat ObjectX from ShapeNet-V2 [61] to simulate classification data. Similarly to ImageNet [47], ShapeNet organizes 3D shapes according to the WordNet hierarchy [62]. From ShapeNet, we select the classes that are also included in the VisDA target dataset (containing real-world images) [24], which are used as our target data. This amounts to 7 classes and 200 models which are randomly selected for each class. During pre-possessing, models in each class are aligned in the same direction and scaled to a uniform size. Table I shows the statistics

### Table I

| Task          | Dataset          | #Models | #Classes (IDs) | Attr |
|---------------|------------------|---------|---------------|------|
| Image cla.    | VisDA Source [24]| 1,907   | 12            | X    |
| Person re-ID  | RandPerson [25]   | 8,000   | 8,000         | X    |
| PersonX [5]   |                  | 3,000   | 3,000         | ✓    |
| Vehicle re-ID | VehicleX         | 1,362   | 1,362         | ✓    |

Note that ObjectX and VehicleX are newly introduced. “Attr” denotes whether a dataset has attribute labels (e.g., orientation). The number of synthetic images here is not compared, because a potentially unlimited number of images can be created by these engines. Of note, for object re-ID, the number of models is equal to the number of IDs.

Fig. 2. Sample 3D models for ObjectX, PersonX [5], and VehicleX [11], which are used for synthesizing images for image classification, person re-ID, and vehicle re-ID, respectively.
of ObjectX, and Fig. 2 visualizes some 3D shapes we collected for categories airplane, bus, skateboard, train, motorcycle, and knife.

We use PersonX to simulate person re-ID data. This asset is introduced by Sun et al. [5] and has 1,266 different manually constructed person models (identities), including 547 females and 719 males. PersonX models are hand-crafted by professional 3D modelers, with a special focus on appearance diversity. To explain, the backbone models of PersonX have various ages, hairstyles, and skin colors. For each backbone model, the clothes are also chosen from a diverse range including T-shirts, skirts, jeans, shorts, pants, slacks, etc. With real-world-like textures, these clothes have good visual authenticity. Furthermore, a person can take various actions (e.g., walking, running) when being filmed. Viewpoint alignment and scaling are performed for 3D objects in PersonX.

VehicleX, introduced in our conference paper [11], is used to simulate data for the vehicle re-ID task. It has a wide range of simulated backbone models and textures and adapts well to the variance of real-world datasets. Specifically, it has 272 backbones which are hand-crafted by professional 3D modelers. The backbones include 11 mainstream vehicle types including sedan, SUV, van, hatchback, MPV, pickup, bus, truck, estate, sportscar and RV. On the basis of these backbones, we obtain 1,362 vehicle identities by adding various colored textures or accessories. A comparison of VehicleX with an existing vehicle re-ID dataset (i.e., PAMIRI [27]) is presented in Table I. VehicleX is three times larger than the synthetic dataset PAMTRI in the number of identities, and, similar to ObjectX and PersonX, can potentially render an unlimited number of images by varying the attributes. In VehicleX, similarly to PersonX, vehicle models are aligned to their frontal views and properly scaled. Note that for object (person and vehicle) re-ID, the testing procedure is retrieval; therefore, synthetic data need not to have the same classes with real-world target data. VehicleX has been used as training data in the AI city challenge in CVPR 2020 and CVPR 2021.1

B. Camera Model

To capture 2D images of the 3D objects, we create a camera model, as illustrated in Fig. 3. Let the world coordinate system be \(O(x, y, z)\), in which the origin is the center of the 3D object, and the \(y\) axis points toward a certain direction of each class of objects (e.g., the frontal view of airplanes). The camera coordinate system \(C\) is denoted by \((x', y', z')\). To ensure that the target is filmed and to simplify the camera model, we hold a prior that the camera is always facing the 3D object, i.e., towards the center of the world coordinate system (or object center). Thus, we define the \(z'\) axis as pointing away from the center of the object.

We proceed to consider extrinsic and intrinsic parameters that transform the world coordinate system into the camera coordinate system. **Extrinsic parameters** are determined by the rotation matrix \(R\) and translation vector \(T\). \(R\) includes object rotation and camera rotation. Object rotation includes the azimuth \(\alpha\) and elevation, which can be represented by the function of camera height \(h\) and camera distance \(d\). Camera rotation involves only in-plane rotation, because we assume that our camera always faces the object. Thus, the rotation matrix \(R\) is a function of \(\alpha\), \(h\), \(d\), and \(\theta\), and can be written as \(R(\alpha, h, d, \theta)\). The translation vector \(T\) is determined directly by camera height \(h\) and camera distance \(d\), and thus can be written as \(T(h, d)\). For **intrinsic parameters**, we use a fixed focus length \(f\) and image resolution \(\gamma\). For example, we set the world coordinate system to a resolution of \(1,920 \times 1,080\) pixels when capturing vehicle images. Given the above extrinsic parameters and intrinsic parameters, the projection matrix can be written as,

\[
P = \begin{bmatrix}
\gamma f & 0 & 0 \\
0 & \gamma f & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
R(\alpha, h, d, \theta)_{3 \times 3} & 0_{1 \times 3} \\
0_{3 \times 1} & T(h, d)_{3 \times 1}
\end{bmatrix}.
\]

Thus, given focal length \(f\) and resolution \(\gamma\), the camera model has 4 remaining attributes to be configured: azimuth \(\alpha\), camera height \(h\), camera distance \(d\) and in-plane rotation \(\theta\). These left 4 attributes determine the diverse object-camera placement, thus allowing us to film a variety of images. This camera model is an improved version from our conference paper [11]. The additionally introduced attribute “in-plane rotation” allows us to more flexibly model objects that do not usually “stand” on the ground plane.

C. Configurable Attributes

Our system has a total of 6 editable attributes, which are considered to be influential on the training set quality and the subsequent testing accuracy. They include azimuth, camera height, camera distance and in-plane rotation, as mentioned in Section III-B, plus two lighting attributes: light direction and light intensity. Several details regarding these attributes are provided below.

- **Azimuth** represents the horizontal viewpoint of an object and takes a value between \(0^\circ\) and \(359^\circ\).
- **In-plane rotation** controls camera rotation on the \(z'\) axis. Its value is also between \(0^\circ\) and \(359^\circ\).

1[Online]. Available: https://www.aicitychallenge.org/
Camera height describes the vertical distance of the camera from the ground, from near (numerical value 0) to far (numerical value 100).

Camera distance determines the horizontal distance between the camera and the object center. This factor, taking values from low (0) to high (100), strongly affects on the imaging resolution (the resolution of the entire camera view is set to 1920 × 1080). Imaging viewpoint is impacted by the joint effects of camera height and distance.

Light direction is part of the environment settings. We assume directional parallel light, and the light direction is modeled from east (0°) to west (180°).

Light intensity is another a critical attribute influencing task performance by creating reflections and shadows. We manually define a reasonable range for intensity, from dark (0) to light (100).

Note that the pre-defined ranges are broad and involves little human assumption. For example, light intensity ranges from fully light to dark. From the ranges, we aim to find their optimal values for effective training dataset construction. Moreover, these 6 attributes (examples shown in Fig. 4) are found important for the three object-centric tasks. For other tasks such as semantic segmentation and object detection, which involve complex environments and multiple objects, other influential attributes, such as object (relative) locations, are influential. In this article, we try to keep our investigation focused and in depth and leave the more complex applications for future work.

D. Image Capturing Process

In this section, we describe how to film the 3D objects after varying these editable attributes. To establish controllability, we build a Unity-Python interface by using the Unity ML-Agents toolkit [63], which creates a data transmission path between Unity and Python. Specifically, given a list of parameters from Python, Unity accordingly sets up the camera and environment, then takes the picture and obtains the bounding box precisely according to the object size. The bounding box is then sent back to Python for further processing. Our API allows users to easily obtain rendered images without expert knowledge of Unity. The source code of this API has been released.²

We also consider background and occlusion setups in our system. Specifically, during the attribute distribution optimization process with attribute descent, we render images with black backgrounds and no occlusions. After the optimization is complete, we generate synthetic training data using the optimized attributes while adding random background images and occlusions. For example, when synthesizing vehicle re-ID training data, we add background images from the CityFlow dataset [64], and occluders such as lamp posts, billboards and trash cans. Our preliminary experiments show this strategy increases the diversity of the synthesized training data, consequently enabling robust training of the task model. For example, when using synthetic data only and targeting vehicleID, training on synthetic images with black backgrounds and attribute descent produces 24.12% mAP. The further incorporation of random backgrounds and occlusions yields a notable increase of +11.21% mAP over the training dataset that uses black backgrounds.

IV. PROPOSED METHOD

A. Attribute Distribution Modeling

We model the distribution of the 6 attributes (Section III-C) using Gaussian Mixture Models (GMM). The primary rationale is that for real-world datasets, these attributes usually follow specific patterns. In the image recognition task, patterns often exist with regard to specific categories. For example, photographs of the category clock are usually taken from the front, not the side or back, where the Gaussian distribution would be helpful. Likewise, in the re-ID task, the camera position is usually fixed, so that the camera height and distance of images captured by a certain camera are generally uni-modal. Moreover, pedestrians and vehicles usually move along predefined trajectories, e.g., footpaths for pedestrians and traffic lanes for vehicles. Because multiple moving directions exist on a single path, the

²[Online]. Available: https://github.com/yorkeyao/VehicleX
object orientations (i.e., the azimuth) of these re-ID data exhibit multi-modal distributions. Of note, this modeling strategy has also been adopted by Ruiz et al. [3]. In addition, using Gaussians and their mixtures to model the distribution of attributes allows us to conveniently analyze and visualize datasets.

Although a GMM is parameterized by its mean and covariance, our preliminary shows that the change in covariance produces a less prominent supervision signal, i.e., has a weaker effect on task accuracy than the mean. This finding is understandable because the mean parameters encode how the majority of images appear and more directly reflect the dataset distribution gap. Although our method can optimize covariance, doing so would significantly increase the search space without notable accuracy gain. Based on the above considerations, we simplify the covariance matrix into a diagonal matrix whose diagonal elements are pre-defined.

Formally, let \( A = [A_1, A_2, \ldots, A_K]^T \in \mathbb{R}^K \) represent attributes discussed in Section III-C, where \( K \) stands for the total number of attributes. \( A_i \in \mathbb{R}, i = 1, \ldots, K \) is the random variable representing the \( i \)th attribute. As discussed in Section III-C, we have \( K = 6 \), where \( A_1, A_2, A_3, A_4, A_5, \) and \( A_6 \) denote azimuth, in-plane rotation, camera height, camera distance, light direction, and light intensity, respectively.

Let \( \theta = [\theta_1, \theta_2, \ldots, \theta_M]^T \) represent all the learnable parameters, where \( M \) is the number of these learnable parameters, and \( \theta_i \in \mathbb{R}, i = 1, \ldots, M \), is a single learnable parameter (i.e., mean, as variance is fixed) of one mixture component. Thus, the \( \theta \) can be viewed as a concatenated vector of parameters of all GMMs. Subsequently, we use \( G(\theta) \) to denote the overall distribution parameterized by \( \theta \), which consists of \( K \) GMMs. For example, we define the distribution of in-plane rotation attribute \( A_1 \) using a GMM consisting of three components. For each component, it has a mean value as a learnable parameter. Formally, they are denoted as \( \theta_1, \theta_2, \) and \( \theta_3 \). For attributes that empirically have simpler distributions, we use GMM with fewer components. For example, we define the camera distance attribute \( A_2 \) to follow a single-component Gaussian, which has only one learnable parameter \( \theta_1 \).

Furthermore, to make things easier for optimization, we let \( S = [S_1, S_2, \ldots, S_M]^T \) represent a collection of search spaces for learnable parameters \( \theta = [\theta_1, \theta_2, \ldots, \theta_M]^T \), where \( S_i \in \mathbb{R}^{d_i}, i = 1, \ldots, M \), is the search space of parameter \( \theta_i \). The dimension \( d_i \) of \( S_i \) is dependent on the attribute \( \theta_i \). For example, we have \( \theta_1, \theta_2, \) and \( \theta_3 \) are learnable parameters of attribute \( A_1 \) (in-plane rotation), and their search spaces \( S_1 = S_2 = S_3 = [30 l, 11]^T \), which is a 12-dim vector including angles between \( 0^\circ \) and \( 330^\circ \) with an increment of \( 30^\circ \). A complete attribute distribution and search space description of all the learnable parameters is provided in Section V-B.

### B. Optimization

To describe the attribute descent optimization process, we introduce the objective function, detail the proposed attribute descent algorithm and then analyze its convergence property.

**Objective function**: Given real-world target data, we aim to synthesize a dataset that has a minimal (content) distribution difference with it. Formally, we denote the sets of synthetic images and real images as \( D^s \) and \( D^r \), respectively, where the synthetic dataset is simulated following the distribution of vector \( A \). We further write \( D^s \) as \( \{ R(a_n) \}_{n=1}^N \), where \( N \) is the number of images in \( D^s \). \( R(\cdot) \) is the underlying rendering function of Unity using attribute vector \( a_n \) as input and producing a synthetic image \( R(a_n) \). Each attribute vector \( a_n \) is sampled from distribution parameterized by \( \theta \), i.e., \( a_n \sim G(\theta) \). We aim to optimize \( \theta \) to minimize the distribution difference measure Fréchet Inception Distance (FID) [10] between \( D^s \) and \( D^r \). Correspondingly, our objective function is written as:

\[
\theta^* = \arg \min_{\theta} \text{FID}(D^s, D^r),
\]

where

\[
D^s = \{ R(a_n) \}_{n=1}^N, a_n \sim G(\theta),
\]

and

\[
\text{FID}(D^s, D^r) = ||\mu^s - \mu^r||^2_2 + Tr \left( \Sigma^s + \Sigma^r - 2(\Sigma^s \Sigma^r)^{\frac{1}{2}} \right).
\]

In (4), \( \mu^s \in \mathbb{R}^d \) and \( \Sigma^s \in \mathbb{R}^{d \times d} \) denote the mean and covariance matrix, respectively, of the image descriptors of the synthetic dataset \( D^s \), and \( \mu^r \) and \( \Sigma^r \) are those of the real-world dataset \( D^r \), respectively. \( d \) is the dimension of the image descriptors, which are extracted by the InceptionV3 model [65] pre-trained on the ImageNet dataset [47].

The benefit of using FID as the loss function is two-fold. First, it does not require ground truths of the target real-world dataset \( D^r \) and instead can be computed using target images only. This obviously saves human labeling efforts. Second, because the FID loss is computed between two sets of images, it is much faster to compute than existing methods using the task loss [6], [66]. The latter requires training the task model (e.g., object detection [67]) with the generated training set and uses the obtained model task performance as loss, which is time-consuming.

It is important to note that the objective function (2, 3), and (4) is non-differentiable with respect to \( \theta \), primarily because the rendering function (through the 3D engine Unity) is not differentiable. As such, we cannot perform optimization by directly using gradient descent.

**Attribute descent algorithm**: We are motivated by coordinate descent, a classical optimization method that can work under derivative-free scenarios [68]. To find a local minimum, it selects a coordinate direction to perform a step-by-step search and iterates among coordinate directions. Compared with grid search that considers the entire search space, coordinate descent has significantly reduced search space and thus running time.

Our loss function is parameterized by \( \theta \) which encodes the distribution of the configurable attributes. Following coordinate descent, we propose attribute descent (Algorithm 1) to iteratively optimize each parameter. Specifically, we view each parameter as a coordinate in coordinate descent. In each iteration, we successively vary the value of a parameter so as to move toward a local minimum of the objective function.
Algorithm 1: Attribute Descent.

1: **Input:** Initialized learnable parameters $\theta = [\theta_1, \ldots, \theta_M]^T$, search space $S = [S_1, S_2, \ldots, S_M]^T$, rendering function $\mathcal{R}(\cdot)$ and target dataset $D^r$.
2: **Hyperparameters:** $J$ epochs for attribute descent and synthetic dataset size $N$.
3: **Begin:**
4: 4: $\text{Optimal} = \infty$; $\triangleright$ Initialize the optimal FID value
5: for $j = 1$ to $J$ do $\triangleright$ Number of epoch $J$
6: for $i = 1$ to $M$ do $\triangleright$ Enumerate parameters
7: for $z \in S_i$ do $\triangleright$ Traverse search space
8: $D^s = \{ \mathcal{R}(a_n)|a_n \sim G([\theta_i^1, \ldots, \theta_i^{j-1}], z, \theta_{i+1}^{-1}, \ldots, \theta_M^{-1})^T)^N_{n=1} \}$ $\triangleright$ Generate synthetic dataset
9: Score $= \text{FID}(D^s, D^r)$ $\triangleright$ FID calculation
10: if Score $<$ Optimal then
11: $\text{Optimal} = \text{Score}$ $\triangleright$ Update FID
12: $\theta_i^j = z$ $\triangleright$ Update parameters
13: $\theta_i^j = z$ $\triangleright$ Update parameters
14: $\theta_i^{j+1} = \theta_i^j$ $\triangleright$ Update parameters

Formally, to iteratively optimize the objective function (2) with respect to $\theta$, we first initialize $\theta$ at epoch 0:

$$\theta^0 = [\theta_1^0, \ldots, \theta_M^0]^T.$$  (5)

Then at the $j$th epoch, we get $\theta^j$ from $\theta^{j-1}$ by iteratively solving the single variable optimization problems. Specifically, at $i$th iteration in epoch $j$, we optimize a single parameter $\theta_i^j$ in $\theta^j$, to its best value $z$ in the search space $S_i$:

$$\theta_i^j = \arg \min_{z \in S_i} \text{FID}(D^s, D^r),$$  (6)

where

$$D^s = \{ \mathcal{R}(a_n)|a_n \sim G([\theta_i^1, \ldots, \theta_i^{j-1}], z, \theta_{i+1}^{-1}, \ldots, \theta_M^{-1})^T)^N_{n=1} \}. $$  (7)

In (6) and (7), an iteration is defined as the duration for which a single parameter $\theta_i^j$, $i = 1, \ldots, M$ undergoes an optimization process (from Step 7 to Step 14 in Algorithm 1). An epoch is defined as the duration for which all parameters undergo one attribute descent round.

In attribute descent, each iteration performs a greedy search for a single parameter while values of the other parameters are fixed. Therefore, each iteration finds the best value for a single parameter, and an epoch gives values for the entire parameter vector $\theta$. In our experiments, the entire optimization process usually converges in 2 epochs.

**Convergence characteristics:** Attribute descent has the following properties in regarding model convergence.

**Fast conditional updates:** The problem configuration allows us to individually optimize parameters quickly. As stated in Section IV, we can update each attribute by using discrete values within a range. For example, the search space for azimuth is between 0° and 330° with a 30° interval. Therefore, only 12 iterations are necessary, thus greatly decreasing the computational time. Note that, using attribute descent for complex tasks such as semantic segmentation would be result in a much slower overall procedure because these tasks have many more attributes to optimize.

**Strong stability:** Attribute descent can steadily converge because of its “one-at-a-time” optimization characteristics. That is, it gradually and greedily updates each parameter if and only if this update can improve the objective function. Compared with “all-at-once” algorithms like gradient descent which requires expert-level training skills to converge, attribute descent is easier to train and more stable in convergence.

**Analysis and visualization of the attribute descent process:** In Fig. 5, we present how task performance (re-ID accuracy) and domain discrepancy metric (FID) change during the attribute descent process. We observe that attributes are successively optimized when the FID value decreases and the re-ID accuracy mAP increases. Specifically, from the slope of the curves in Fig. 5(a), when optimizing attributes in the order of “orientation” -> “lighting” -> “camera pose”, we observe that after orientation attributes (i.e., in-plane rotation and azimuth) are optimized, a large decrease in FID occurs from 147.85 to 91.14 and a large mAP increase occurs from 12.1% to 21.94%. Subsequently, after lighting attributes are optimized, we have -7.2 FID and +10.7% mAP. The optimization of the camera attributes leads to -4.11 FID and +2.4% mAP. These observations illustrate that all the attributes are useful for improving the training data quality. In addition, in Fig. 5(b), the content of generated images becomes increasingly similar to the target real images through the optimization procedure, thus suggesting the effectiveness of attribute descent.

C. Application Scenarios

1) Training With Synthetic Data Only: Setting: In the first application, we use generated synthetic data to replace real-world data for task model training. Specifically, we perform attribute descent to optimize synthetic data toward target data without labels, train task model on the adapted synthetic data, and then test the models on target test sets.

**Obtaining target class or camera labels:** As stated in Section IV-A, attribute descent requires known class or camera labels, because we observe different attribute distributions for different categories or cameras; examples are shown in Section V-D.

In mining class or camera labels from the unlabeled target domain, we leverage pseudo labels for classification. Specifically, we use the model trained on synthetic data (i.e., ObjectX) for pseudo label assignment with random attributes. Table II shows that the classification model trained on ObjectX achieves 65.0% average top-1 accuracy on the VisDA test set; therefore, the pseudo labels are relatively reliable. After we obtain the pseudo labels for the target domain, we perform attribute descent for each class by following Algorithm 1. Because ObjectX includes 7 classes, we optimize 7 attribute lists for the classification task.

For object re-ID, we simply perform attribute descent against each camera in the target domain, in order to simulate images
with similar content to those from each camera. For example, we optimize 6 and 20 attribute lists for the Market and VeRi (both are target domains) training sets which have 6 and 20 cameras, respectively. Note that assuming knowledge of the camera label is a common practice in unsupervised domain adaptive re-ID [59].

2) Augmenting Target Training Data: Setting: For this application, we use synthetic data to augment real-world training data. Specifically, labels of the target data are provided in both attribute descent and task model training. We combine the adapted synthetic data and real-world data and perform two-stage joint training [70] to obtain task models. For the classification task, when labels of the target training set are given, we optimize attributes for each category directly. Likewise, for the re-ID task, when labels for each camera are given, we optimize attributes for each camera.

Two-stage training [70] is conducted in training data augmentation wherein synthetic and real-world data are both used in training. We mix synthetic data and real-world data in the first stage and finetune with real-world data only in the second stage. In CityFlow, for example, in the first stage, we train on both real and synthetic data, where we classify vehicle images into one of the 1,695 (333 real + 1,362 synthetic) identities. In the second stage, we replace the classification layer with a new classifier fine-tuned on the real dataset (333 classes). When conducting the second stage training, we have a lower learning rate than that in the first stage, with details following [70].

3) Understanding Dataset Content Numerically: Attribute descent provides a numerical way to understand (and sometimes visualize) the content of datasets. Specifically, given a certain category (or camera), we use attribute descent and synthetic models to build a proxy set, which has similar content distribution. After optimization, we can obtain the values of an attribute of interest for each image, and collectively obtain the value distribution for a dataset. We can then use either statistics or visualization tools to understand a certain aspect of the content of a dataset through the attribute. For example, we can visualize the viewpoint distribution in a 3D sphere (as shown in Section V-D). We can also use histograms to present the distribution of lighting intensity of a dataset.

V. EXPERIMENT

We evaluate the effectiveness of attribute descent on image classification, person re-ID and vehicle re-ID. In all tasks, given target data (Section V-A), we use attribute descent to synthesize a training set that has similar attribute distributions.

A. Source and Target Datasets

Image classification: We use ObjectX (described in Section III-A) as the source and the VisDA [24] target set as the target domain. The original VisDA target set has 12 classes of real-world images. Among the 12 classes, we select 7 that are also included in ShapeNet V2 [60] and thus ObjectX. A total of 33,125 images are present in the 7 classes. For each class, the ratio of the number of training images to that of testing images is 1:7 or 1:1.

Person re-ID: We use PersonX as a source, and use two real-world datasets as target: Market-1501 (denoted as Market) [51] and DukeMTMC-reID (denoted as Duke) [52]. Market has 1,501 IDs, 12,936 training images and 19,732 gallery images filmed by 6 cameras. A total of 751 out of 1,501 IDs are used for training and the remaining 750 are used for testing. The query set includes 3,368 bounding boxes from 750 identities. Duke contains 1,404 IDs and 36,441 images captured by 8 cameras. There are 16,522 images from 702 identities for training, 2,228 query images from another 702 identities and 17,661 gallery images for testing.

Vehicle re-ID: Apart from the synthetic VehicleX dataset, we use three real-world vehicle re-ID datasets as target domain data. VehicleID [8] contains 222,629 images of 26,328 identities. Half the identities are used for training, and the other half are used for testing. Three test splits exist: “Small”, “Medium” and “Large”, representing the number of vehicles in the test set. Specifically, “Small” has 800 vehicles and 7,332 images, “Medium” has 1,600 vehicles and 12,995 images, and “Large” has 2,400 vehicles and 20,038 images. The VeRi-776 dataset [9] contains...
Fig. 6. Examples of synthesized images for person re-ID (left) and vehicle re-ID (right). A1 and A2: images simulated by random attributes. B1 and B2: image simulated by attributes optimized through attribute descent, to approximate the distribution of real-world target data E1 and E2, respectively. C1 and C2: we apply SPGAN [69] to translate images in A1 and A2, respectively into the style of the target domain. D1 and D2: SPGAN is applied to images in B1 and B2, respectively. We can observe that attribute descent changes the image content such as camera viewpoint and object orientation, while SPGAN amends image styles. The two forces are complementary to each other, as shown in our experiment.

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. We use “VeRi” for short in what follows. CityFlow-reID [64] has more complex environments and has 40 cameras in a diverse environment where 34 of them are used in the training set. The dataset has in total 666 IDs where half are used for training and the rest for testing. We use “CityFlow” for short in the following context.

Evaluation protocol: For image classification, we report the top-1 accuracy averaged over all the categories. For object re-ID, we use mean average precision (mAP) and cumulative match curve (CMC) scores to measure system accuracy, e.g., “Rank-1” and “Rank-5”. “Rank-1” denotes the success rate of finding the true match in the first rank, and “Rank-5” means the success rate of ranking at least one true match within the top 5.

B. Experimental Details

Attribute descent settings: As discussed in Section IV-A, we model the distribution of the 6 attributes using GMM. Specifically, When using GMM, we set the number of Gaussian components to 3, 6, 1, 1, 1, and 1 for in-plane rotation, azimuth, light intensity, light direction, camera height, and camera distance, respectively. Meanwhile, as mentioned in Section IV-A, only means of the Gaussians are optimized, and initialized from the lowest value in the search space. For each learnable mean value \( \theta_i, i = 1, \ldots, M \) in \( \theta \), the search space is specified in the range defined in Section III-C, and their search steps are 12, 12, 10, 6, 10, and 5 for in-plane rotation, azimuth, light intensity, light direction, camera height, and camera distance, respectively.

The covariance matrices are diagonal matrices with predefined diagonal elements. Specifically, the diagonal elements are (10, 10, 10), (20, 20, 20, 20, 20, 20), (0.63), (7.07), (0.4) and (0.6) for in-plane rotation, azimuth, light intensity, light direction, camera height, and camera distance, respectively. In attribute descent, the number of epochs of attribute descent is 2, which usually leads to convergence.

Image style transformation: For object re-ID (both person and vehicle), we use SPGAN [14] to transfer the appearance of synthetic data to that of the target domain. For image classification, we do not use style transformation.

Task model configuration. For the classification task, we use ResNet-50 [49] to classify the 7 classes. For person re-ID, we use multiple task models, including ID-discriminative embedding (IDE) [73], the part-based convolution (PCB) [74], and TransReid [76]. Note that when implementing these task models, we use their official implementations with default hyperparameters including learning rate and training epochs. For IDE, we adopt the strategy from [80] which uses ResNet-50 [49] and adds batch normalization and removes ReLU after the final feature layer. For PCB, we use the ResNet-50 backbone and vertically partitions an image into six equal horizontal parts. For vehicle re-ID, we also use IDE and PCB. For IDE, we also use the ResNet-50 backbone. For PCB, we use the ResNet-50 backbone and horizontally divide an image into six parts.

C. Quantitative Evaluation of Attribute Descent

Given a target set, attribute descent allows us to synthesize a dataset that has a similar distribution on the content (attribute) level. In this section, we demonstrate three application scenarios of the synthesized data in object-centric tasks: training with synthetic data only, real-synthetic data augmentation, and dataset
TABLE II

| Application         | Training data | Type | Val-test split | Model  | plane  | bus   | car   | knife | moped  | skis   | trans | per-class |
|---------------------|---------------|------|----------------|--------|--------|-------|-------|-------|--------|--------|-------|-----------|
| VisDA Source        | S             | 1:7  | ResNet-50      | 72.13  | 38.57  | 52.20 | 40.89 | 91.21 | 62.35  | 57.99  | 75.65 | 65.04     |
| Synthetic Training  |               |      |                |        |        |       |       |       |        |        |       |           |
| OX (Ran. Attr.)     | S             | 1:7  | ResNet-50      | 77.95  | 48.06  | 62.33 | 40.88 | 92.11 | 57.99  | 75.65  | 65.04 | 65.00     |
| OX (Attr. Desc.)    | S             | 1:7  | ResNet-50      | 83.01  | 51.01  | 60.11 | 45.65 | 82.60 | 59.36  | 72.79  | 58.38 | 70.24     |

We use VisDA target as the target domain (real-world), and ObjectX ("OX") as the source. In terms of data composition, "S" represents synthetic data only, "R" denotes real-world data only, and "R+S" means both synthetic data and real-world data are used. Two validation-test splits are used for data augmentation, i.e., 1:1 and 1:7.

TABLE III

| Application         | Training data | Type | Val-test split | Model  | plane  | bus   | car   | knife | moped  | skis   | trans | per-class |
|---------------------|---------------|------|----------------|--------|--------|-------|-------|-------|--------|--------|-------|-----------|
| VisDA Target        | R             | 1:7  | ResNet-50      | 96.24  | 85.69  | 91.47 | 94.33 | 92.86 | 90.88  | 90.31  | 91.6  | 66.45     |
| VisDA Target+OX (Ran. Attr.) | R+S  |      | ResNet-50      | 96.90  | 89.86  | 94.33 | 95.04 | 92.28 | 92.38  | 92.07  | 93.6  | 78.04     |
| VisDA Target+OX (Attr. Desc.) | R+S  |      | ResNet-50      | 98.13  | 89.38  | 95.96 | 96.82 | 95.34 | 93.44  | 93.91  | 94.70 | 95.5      |

We use VisDA target as the target domain (real-world), and ObjectX ("OX") as the source. In terms of data composition, "S" represents synthetic data only, "R" denotes real-world data only, and "R+S" means both synthetic data and real-world data are used. Two validation-test splits are used for data augmentation, i.e., 1:1 and 1:7.

TABLE IV

| Application         | Training data | Type | Val-test split | Model  | plane  | bus   | car   | knife | moped  | skis   | trans | per-class |
|---------------------|---------------|------|----------------|--------|--------|-------|-------|-------|--------|--------|-------|-----------|
| Duke                | R             | 1:7  | IDE [74]       | 7.38   | 14.55  | 1.92  |       |       |        |        |       |           |
| Market              | R             | 1:7  | IDE [74]       | 12.17  | 31.68  | 18.07 |       |       |        |        |       |           |
| PTX (Ran. Attr.)    | S             | 1:7  | IDE [74]       | 17.01  | 33.49  | 6.30  |       |       |        |        |       |           |
| PTX (Attr. Desc.)   | R             | 1:7  | IDE [74]       | 34.71  | 51.60  | 15.01 |       |       |        |        |       |           |

We use market as the target and use PersonX ("PX") as the source. A few state-of-the-art re-ID models are used. Data type notations are the same as those in Table II. MAP (%) and CMC scores (%) are reported. The bold values mean the highest task accuracy achieved when using synthetic data to augment the target training data.

From VehicleX to VeRi, attribute descent again contributes to a +7.69% improvement in Rank-1 accuracy.

Of note, accuracy under this application scenario is usually lower than that of the state of the art or that produced by in-distribution training sets. This difference is understandable, because synthetic data have a relatively low resemblance with respect to the target data in terms of appearance.

Effectiveness of optimized synthetic data in augmenting the target training data: After optimizing attributes to mimic the target domain, we mix the generated synthetic data with the target training data (with labels) to train the recognition / re-ID models. Apart from the two-stage training strategy (see Section IV-C2), no additional training skills are employed. We again compare attribute descent with random attributes. Experimental results for the three object-centric tasks are summarized across Tables II, III, IV, VIII, VI, and VII, respectively.

From these results, we observe that when using only synthetic data for training, the data generated from learned attributes achieve much higher task accuracy than those generated from random attributes. For example, when adapting ObjectX to VisDA, attribute descent results in a +4.61% improvement in per-class accuracy over using random attributes. When adapting PersonX to Market, attribute descent yields a +7.69% improvement in Rank-1 accuracy over using random attributes.

We use VisDA target as the target domain (real-world), and ObjectX ("OX") as the source. In terms of data composition, "S" represents synthetic data only, "R" denotes real-world data only, and "R+S" means both synthetic data and real-world data are used. Two validation-test splits are used for data augmentation, i.e., 1:1 and 1:7.

Effective visualization. In each application scenario, we compare attribute descent with several existing methods.

Effectiveness of attribute descent for synthetic-data-only training: After generating content-adapted synthetic data with attribute descent (visual examples in Fig. 6), we train the subsequent classification / re-ID models with generated synthetic data only. We mainly compare the optimized synthetic dataset with those generated with random attributes, a commonly used baseline in the community [4], [66]. Here, “random” means that attributes follow the uniform distribution, where their value ranges are the same as the search space of learned attributes. Experimental results on the VisDA target set, Market, Duke, VehicleID, VeRi and CityFlow are shown in Tables II, III, IV, VIII, VI, and VII, respectively.

From these results, we observe that when using only synthetic data for training, the data generated from learned attributes achieve much higher task accuracy than those generated from random attributes. For example, when adapting ObjectX to VisDA, attribute descent results in a +4.61% improvement in per-class accuracy over using random attributes. When adapting PersonX to Market, attribute descent yields a +7.69% improvement in Rank-1 accuracy over using random attributes.

Effective visualization. In each application scenario, we compare attribute descent with several existing methods.

Effectiveness of attribute descent for synthetic-data-only training: After generating content-adapted synthetic data with attribute descent (visual examples in Fig. 6), we train the subsequent classification / re-ID models with generated synthetic data only. We mainly compare the optimized synthetic dataset with those generated with random attributes, a commonly used baseline in the community [4], [66]. Here, “random” means that attributes follow the uniform distribution, where their value ranges are the same as the search space of learned attributes. Experimental results on the VisDA target set, Market, Duke, VehicleID, VeRi and CityFlow are shown in Tables II, III, IV, VIII, VI, and VII, respectively.

From these results, we observe that when using only synthetic data for training, the data generated from learned attributes achieve much higher task accuracy than those generated from random attributes. For example, when adapting ObjectX to VisDA, attribute descent results in a +4.61% improvement in per-class accuracy over using random attributes. When adapting PersonX to Market, attribute descent yields a +7.69% improvement in Rank-1 accuracy over using random attributes.

Effectiveness of optimized synthetic data in augmenting the target training data: After optimizing attributes to mimic the target domain, we mix the generated synthetic data with the target training data (with labels) to train the recognition / re-ID models. Apart from the two-stage training strategy (see Section IV-C2), no additional training skills are employed. We again compare attribute descent with random attributes. Experimental results for the three object-centric tasks are summarized across Tables II, III, IV, VIII, VI, and VII. Under this application, we also observe consistent improvement brought by the additional synthetic data. For example, from ObjectX to VisDA, the improvement in learned attributes over random attributes is +1.8% in top-1 recognition accuracy. From PersonX to Market and from VehicleX to VeRi, the improvements in mAP are +2.61% and +1.23%, respectively. The improvements appear numerically smaller than those in the “training with synthetic only” setting, because the latter sits on a relatively low baseline due to its
We examine the use case of training data augmentation. Data type annotations and evaluation metrics are the same as those in Table III. "Small", "medium" and "large" refer to the three test splits of the VehicleID test set [8]. "VX" denotes the vehicleX dataset. "VID" means the VehicleID dataset.

The bold values mean the highest task accuracy achieved when using synthetic data to augment the target training data.

![Table V](image)

**Comparison of Various Training Sets When VehicleX is the Source Domain and VehicleID is the Target Domain**

| Training data | Type     | Model | Small Rank-1 | Small Rank-5 | Small mAP | Medium Rank-1 | Medium Rank-5 | Medium mAP | Large Rank-1 | Large Rank-5 | Large mAP |
|---------------|----------|-------|--------------|--------------|-----------|---------------|---------------|-------------|--------------|--------------|-----------|
| VID           | R        | RAM [78] | 79.2          | 91.3         | 72.3      | 87.0          | 67.7          | 84.5        | -            | -            | -         |
|               |          | AAVER [53] | 74.69         | 93.82        | 68.62     | 89.95         | 63.54         | 85.64       | -            | -            | -         |
|               |          | GSTE [79] | 75.9          | 84.2         | 74.8      | 83.6          | 74.0          | 82.7        | 72.4         | 78.51       | -         |
| VID+VX       | R+S      | IDE [74] | 80.2          | 93.98        | 76.94     | 90.84         | 73.45         | 86.66       | 80.55        | -            | -         |
| VID+VX (Att. Desc.) | R+S | 81.50        | 94.85         | 87.33        | 77.62     | 92.20         | 74.87         | 89.90       | 81.35        | -            | -         |

Data type annotations and evaluation metrics are the same as those in previous tables. The IDE and PCB baseline models are evaluated,

The bold values mean the highest task accuracy achieved when using synthetic data to augment the target training data.

![Table VI](image)

**Comparison of Various Training Sets When VeRi is the Target Domain**

| Appl. Training | Type | Model | Small Rank-1 | Small Rank-5 | Small mAP | Medium Rank-1 | Medium Rank-5 | Medium mAP | Large Rank-1 | Large Rank-5 | Large mAP |
|----------------|------|-------|--------------|--------------|-----------|---------------|---------------|-------------|--------------|--------------|-----------|
| ImageNet      | R    | IDE   | 30.57        | 40.85        | 8.19      | 95.98         | 20.32         | 10.39       | 96.65        | 17.19        | 99.48     |
| VehicleID     | R    | IDE   | 59.24        | 71.16        | 56.34     | 89.97         | 71.88         | 56.65       | 80.94        | 71.88        | 80.94     |
| CityFlow      | R    | IDE   | 69.56        | 81.35        | 26.71     | 73.75         | 21.29         | 73.75       | 72.4         | 79.51       | 72.4     |
| VX (Rand. Attr.) | S   | IDE [74] | 43.56        | 61.98        | 13.85     | 51.25         | 67.70         | 21.29       | 51.25        | 67.70        | 21.29     |
| VX (Att. Desc.) | S   | IDE [74] | 43.56        | 61.98        | 13.85     | 51.25         | 67.70         | 21.29       | 51.25        | 67.70        | 21.29     |

Data type annotations and evaluation metrics are the same as those in previous tables. The IDE and PCB baseline models are evaluated.

The bold values mean the highest task accuracy achieved when using synthetic data to augment the target training data.

![Table VII](image)

**Comparison of Various Training Sets When CityFlow is the Target Domain**

| Training data | Type | Model | Small Rank-1 | Small Rank-20 | Small mAP | Medium Rank-1 | Medium Rank-20 | Medium mAP | Large Rank-1 | Large Rank-20 | Large mAP |
|---------------|------|-------|--------------|---------------|-----------|---------------|---------------|-------------|--------------|--------------|-----------|
| CityFlow      | R    | IDE   | 94.62        | 80.04         | 25.01     | 94.62         | 80.04         | 25.01       | 94.62        | 80.04        | 25.01     |
|               |      | IDE (CE+Tr.) | 78.80       | 78.80         | 25.57     | 78.80         | 78.80         | 25.57       | 78.80        | 78.80        | 25.57     |
| CityFlow+VX   | R+S  | IDE [74] | 56.75        | 72.24         | 30.21     | 56.75         | 72.24         | 30.21       | 56.75        | 72.24        | 30.21     |
| CityFlow+VX (Rand. Attr.) | R+S   | IDE [74] | 59.77        | 80.13         | 33.81     | 59.77         | 80.13         | 33.81       | 59.77        | 80.13        | 33.81     |
| CityFlow+VX (Att. Desc.) | R+S  | IDE (CE+Tr.) | 63.59        | 82.69         | 35.96     | 63.59         | 82.69         | 35.96       | 63.59        | 82.69        | 35.96     |
|               |      | IDE [74] | 64.07        | 83.27         | 37.16     | 64.07         | 83.27         | 37.16       | 64.07        | 83.27        | 37.16     |

Data type annotations and evaluation metrics are the same as those in previous tables. The IDE and PCB baseline models are evaluated.

The bold values mean the highest task accuracy achieved when using synthetic data to augment the target training data.

appearance discrepancy between source the target data. To summarize, the superiority of attribute descent over random attributes is also shown in Fig. 7.

In image classification, we evaluate two ratios (i.e., 1:1 and 1:7) of the number of training data to that of test data. Table II indicates that our method brings consistent superiority to training with target data only and augmentation with randomly synthesized data under both ratios. When using a 1:1 ratio, the improvement is smaller in magnitude, a finding that is understandable because using more training data would lead to higher baseline accuracy.

In object re-ID, we also demonstrate the benefit of synthetic data augmentation to a few existing models. For example, in Table III, when using IDE, PCB and TransReid architectures, augmenting the training set with optimized synthetic data is consistently beneficial compared with using real data only. Similar observations are made on the PersonX to Duke setting (Table IV) and the VehicleX to VeRi setting (Table VI).

**Comparison with existing gradient-free methods:** We compared the proposed attribute descent with random search, evolutionary algorithm (i.e., genetic algorithm), Bayesian optimization, and reinforcement learning (i.e., LTS). These methods have been used as a strong baseline in hyper-parameter search and neural architecture search [81], and only reinforcement learning has been previously used for content-level domain adaptation [3].

Specifically, for the random search, we randomly sample attribute values 200 times and choose the attribute list with the best FID score. For the evolutionary algorithm, we use a generic algorithm with a fitness function equal to FID [82]. For Bayesian optimization, we use the pipeline stated in [83]. For reinforcement learning, we reproduce the LTS structure [3] and replace the task loss with the FID score. When comparing these methods, we use the same distribution definition and initialization as attribute descent. For a fair comparison, we report the best results after 200 iterations of training (i.e., computing the FID score 200 times and taking the lowest FID score). Comparison results are shown in Fig. 8 and Table VIII, from which four observations are made.

First, under the same task network IDE [73], learned attributes (regardless of which optimization method is used) outperform random attributes in both FID and mAP, demonstrating the
and are usually filmed vertically (in a normal orientation). This with real data only by +0.52% in Rank-1 accuracy. On the VehicleID dataset, the mAP of our system exceeds TransReID by +0.51% and +5.6%.

| Training data | Type | FID | Rank-1 | Rank-5 | mAP |
|---------------|------|-----|--------|--------|-----|
| VeRi          | R    | 45.39 | 28.00 | 41.11 | 38.59 |
| Cityflow      | R    | 75.36 | 38.23 | 53.70 | 45.57 |
| VX (Ran. Att.)| S    | 134.75 | 18.76 | 30.11 | 22.00 |
| VX (Ran. Sear.)|     | 109.94 | 21.84 | 35.25 | 26.35 |
| VX (Evo. Algo.)|    | 105.14 | 21.97 | 35.78 | 27.12 |
| VX (Bay. Opt.)|     | 99.64 | 22.57 | 38.15 | 30.05 |
| VX (LTS)      |     | 95.27 | 24.03 | 38.62 | 32.21 |
| VX (Attr. Desc.)|   | 77.96 | 28.04 | 41.85 | 35.33 |

The training set comparison when VehicleX is the source domain and VehicleID is the target domain, under training with synthetic data only. The IDE [74] model and the “large” train-test split is used. Lower FID indicates lower domain discrepancy with VehicleID.

Impact of different attributes: We perform ablation studies on each group of attributes: object orientation, camera pose and lighting. Results on the application of training with synthetic data only, and tasks of classification, person re-ID, and vehicle re-ID are summarized in Fig. 11. The results provide us with interesting insights regarding the importance of different attributes in these tasks. First, we observe that all three groups of attributes are necessary for good optimization results, where omitting any of them would decrease the accuracy. For example, when we use random values for orientation attributes, task accuracy drops by 15.7% and 9.95% in top-1 recognition rate and mAP on classification and person re-ID task, respectively. Second, we find that attributes have different importance. Specifically, orientation attributes are the most important. For example, in the vehicle re-ID task, not optimizing vehicle orientation leads to a -10.52% drop in mAP, while is much more than the drop caused by omitting camera pose (-3.54%) and lighting (-1.6%).

D. Numerically Understanding Dataset Content

This section uses viewpoint as an example to showcase the application of attribute descent in numerically understanding dataset content. As shown in Figs. 9 and 12, after performing attribute descent on the corresponding synthetic assets, we plot obtained viewpoint value distributions on the unit sphere, where each point on the unit sphere represents a camera pointing toward the center of sphere. The blue points show in-plane rotation of < 30°. The orange points indicate in-plane rotation of > 30°.

Viewpoint Distribution and Bias for Various Classes in VisDA: In Fig. 9 (right) and Fig. 12(a), we observe a significant viewpoint bias in the five categories. For example, we find that airplane, car and bus are usually filmed vertically (in a normal erect position), because the in-plane rotation angles learned for the three class are usually less than 30 degrees. In contrast, knife and skateboard are often filmed from a certain oblique angle with significant in-plane rotations. Moreover, when capturing airplane, bus and skateboard images, the camera is usually at

Benefit of attribute learning for alleviating content difference. Second, random search does not perform well in a limited search time. In fact, it is shown that random search is more effective when many unimportant parameters exist [81]. But in our search space, all the attributes significantly contribute to the distribution differences as shown in Figs. 5 and 11. Third, the evolutionary algorithm, Bayesian optimization, and LTS appear to fall into an inferior local optimum and thus do not produce a lower FID score than attribute descent. To empirically understand such difference, we find that synthetic data optimized by LTS in mimicking the VehicleID dataset exhibit either the car front or rear, whereas VehicleID actually contains both car front and rear. In comparison, our method can sense both directions as it can iterate the entire search space. Fourth, shown in Fig. 8, compared with existing gradient-free methods, attribute descent has the benefit of stable convergence due to its greedy search nature. Given these benefits, attribute descent presents itself as a straightforward yet effective baseline for syn2real content-level domain adaptation.

Positioning among state-of-the-art systems: This article aims to demonstrate the consistent improvement gained through the use of optimized synthetic data, instead of focusing on achieving a new state of the art. Nevertheless, the system augmented with learned synthetic data has very competitive accuracy. Comparisons with several representative state-of-the-art methods are summarized in Tables III, IV, V, VI, and VII. For example, when Market is jointly trained with personX, our system outperforms TransReID [76] with real data only by +0.52% in Rank-1 accuracy. Similarly, on Duke and VehicleID (small), our system exceeds TransReID [76] and GSTE [78] by +0.51% and +5.6% in Rank-1 accuracy. On the VeRi-776 dataset, the mAP of our system using the PCB backbone is 74.51%, which is +2.63% higher than that using PAMTRI [27].

Impact of attribute order in attribute descent: In Fig. 8, we investigate the dependency among attributes by testing whether the order of attributes matters in attribute descent. Using different attribute orders in attribute descent optimization and comparing the FID scores between generated data and target data at epoch I and epoch II, yields two observations. First, after the first epoch, some orders generate lower FIDs than others. For example, the order “orientation -> lighting -> camera pose” results in lower FID than “lighting -> camera pose -> orientation”. This difference is because orientation accounts for more of the discrepancy between synthetic data and real data than camera pose and lighting. Second, although different orders may give different FID values after epoch I, their FID values (and accuracy, not shown in this figure) become similar after epoch II. This property is associated with the coordinate descent algorithm, wherein the order of coordinates in the optimization does not affect final performance.

Impact of different attributes: We perform ablation studies on each group of attributes: object orientation, camera pose and lighting. Results on the application of training with synthetic data only, and tasks of classification, person re-ID, and vehicle re-ID are summarized in Fig. 11. The results provide us with interesting insights regarding the importance of different attributes in these tasks. First, we observe that all three groups of attributes are necessary for good optimization results, where omitting any of them would decrease the accuracy. For example, when we use random values for orientation attributes, task accuracy drops by 15.7% and 9.95% in top-1 recognition rate and mAP on classification and person re-ID task, respectively. Second, we find that attributes have different importance. Specifically, orientation attributes are the most important. For example, in the vehicle re-ID task, not optimizing vehicle orientation leads to a -10.52% drop in mAP, while is much more than the drop caused by omitting camera pose (-3.54%) and lighting (-1.6%).
Fig. 9. Viewpoint distribution visualization for VehicleID and class knife in VisDA. (Left:) by estimating the orientation parameters for each vehicle, our method shows that the viewpoint distribution on VehicleID is bi-modal, where orientations are concentrated in the A front or B rear. (Right:) our visualization method shows that the viewpoint of the class knife on VisDA exhibits various in-plane rotations. The blue dots indicate image samples with in-plane rotation < 30° while the orange dots mean in-plane rotation > 30°.

Fig. 10. Comparison of different attribute orders in attribute descent optimization. We show FID values between generated data and the VehicleID dataset after epoch I and II in attribute descent, which is performed for a total of two epochs (see Algorithm 1). Various optimization orders of attributes are tested. Each order is described by icons representing attributes of object orientation (azimuth and in-plane rotation), camera pose (camera height and distance) and lighting (light direction and intensity). Under different attribute orders, we observe similar FID values after epoch II.

Fig. 11. Ablation studies of each group of attributes: object orientation, camera pose and lighting. Each ablation experiment leaves a certain group of attributes unoptimized (i.e., following the uniform distribution) and is compared with the full system. Top-1 accuracy (%), mAP (%) and mAP (%) are reported on image classification, person re-ID and vehicle re-ID tasks, respectively. We use VisDA, Market, and VehicleID as the target domain for the three tasks, respectively.

the same height as the object, but for knife, it is usually as either a higher or lower position.

Viewpoint Distribution and Bias for Different Cameras in re-ID Datasets: In Fig. 12(b)–(e), we observe very different viewpoint patterns of different cameras. On the Market dataset, the viewpoint distribution for camera 2 is distinct from that for camera 3. Specifically, we observe that camera 2 is higher than camera 3 and the azimuth of camera 2 covers a broader range than that of camera 3. Likewise, on the Duke dataset, camera 4 mainly films people from front or rear angles, whereas camera 7 films from nearly all directions of the azimuth. Similar phenomena are also observed in vehicle re-ID datasets. In VeRi, for example, in contrast to camera 8, which films only car fronts and car rears, camera 7 mainly films vehicles from the side. In summary, significant viewpoint bias exist for different cameras in re-ID datasets. Such bias comes from the fact that camera positions are usually fixed and that objects (person or vehicle) regularly follow predefined lanes. Bias among cameras inevitably leads to bias between datasets, and potentially decrease accuracy when deploying models.

VI. DISCUSSION

Optimization Under Non-Differentiable Simulation Functions: Our system is non-differentiable because of the Unity rendering function. Under this circumstance, the gradient can be estimated by a few existing methods such as finite-difference [6] and reinforcement learning [3]. These methods are best applied in scenarios with a relatively large number of parameters (at least hundreds or thousands) to optimize. However, in object-centric tasks, far fewer attributes to optimize, so we instead propose a stable and efficient optimization approach.

The Relationship Between Distribution Shift and Task Accuracy has been examines in several recent studies. Deng et al. assume a fixed training set and quantitatively measure the strong negative correlation between accuracy and distribution shift of the test set in image classification [84], [85]. In comparison, we assume a fixed test set instead and use the negative correlation in method design: a better training set would have a smaller distribution shift from the test set. This assumption is verified in both object classification and re-ID, which complements [84], [85] from both the assumption and application perspectives.

Can we Use Metrics Other Than FID to Provide Supervision Signals? Two other methods could potentially be used to measure distribution gaps: building a discriminator with an adversarial loss or training a task network with the task loss. However, in
Fig. 12. Viewpoint distribution visualization for different categories in VisDA and different cameras in Market, Duke, VeRi and CityFlow. In image classification (VisDA), we observe that each object has a specific and unique viewpoint pattern. Likewise, each camera in the re-ID task has a distinct pattern. These patterns in classes or cameras reflect the viewpoint bias of datasets.

Beyond object-centric tasks, we experimentally show that attribute descent is also useful in the semantic segmentation task where the street scenes have more complex distributions. For this application, we use the 3D assets (i.e., SceneX) collected in our previous work [29], where 23 controllable attributes are defined, including scene layout, illumination, etc. This is significantly more than the 6 attributes defined in the object-centric tasks. Using the Cityscapes dataset [86] as target and DeepLab-v2 segmentation model [87], we present quantitative results in Fig. 13. In spite of the more challenging setup, we find that our method still maintains its superiority to random attributes. For example, under synthetic only training, attribute descent yields +6.42% improvement in mean intersection over union (mIoU) over the use of random attributes. That said, inheriting from coordinate descent, attribute descent may have lower running efficiency in a complex scenario [88], where we speculate that global optimization algorithms like reinforcement learning will be good alternatives.

VII. CONCLUSION

This article studies how to improve training data quality from the perspective of reducing the domain gap between synthetic...
data and real data on the content level. Specifically, we propose an attribute descent algorithm that can automatically edit the source domain (synthetic) image content in a graphics engine to generate training data with good resemblance to the target domain (real-world). We evaluate this method on object-centric tasks, in which the usage of object bounding boxes decreases the number of attributes to be optimized. Fewer attributes of interest allow us to optimize attributes individually using the proposed attribute descent approach. We show that data synthesized from learned attributes improve task accuracy in two application scenarios: training with synthetic data only and augmenting target data with synthetic data. In addition, using viewpoint as an example, we show that attribute descent enables understanding of the dataset content by computing the attribute distribution equilibrium. We evaluate this method on object-centric datasets, in which the usage of object bounding boxes decreases the number of attributes to be optimized. Fewer attributes of interest allow us to optimize attributes individually using the proposed attribute descent approach. We show that data synthesized from learned attributes improve task accuracy in two application scenarios: training with synthetic data only and augmenting target data with synthetic data. In addition, using viewpoint as an example, we show that attribute descent enables understanding of the dataset content by computing the attribute distribution equilibrium.

Fig. 13. Comparison of training sets synthesized from learned attributes by attribute descent and random attributes, with application to street scene semantic segmentation. We aim to approximate target distribution of street scenes. Similar to Fig. 7, two application scenarios are evaluated: training with synthetic data only (“S”) and training data augmentation (“R+S”). Statistical significance analysis is conducted.

REFERENCES

[1] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 102–118.

[2] C. Sakaridis, D. Dai, and L. Van Gool, “Semantic foggy scene understanding with synthetic data,” Int. J. Comput. Vis., vol. 126, pp. 973–992, 2018.

[3] N. Ruiz, S. Schulter, and M. Chandraker, “Learning to simulate,” in Proc. Int. Conf. Learn. Representations, 2019.

[4] J. Tremblay et al., “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2018, pp. 1082–10828.

[5] X. Sun and L. Zheng, “Dissecting person re-identification from the viewpoint of viewpoint,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 608–617.

[6] A. Kar et al., “Meta-sim: Learning to generate synthetic datasets,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 4550–4559.

[7] J. Hoffman et al., “CyCADA: Cycle-consistent adversarial domain adaptation,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1998–1998.

[8] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, “Deep relative distance learning: Tell the difference between similar vehicles,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2167–2175.

[9] 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, 2016, pp. 1–6.

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local nash equilibrium,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 6629–6640.

[11] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, “Simulating content consistent vehicle datasets with attribute descent,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 775–791.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2242–2251.

[13] A. Shirinastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, “Learning from simulated and unsupervised images through adversarial training,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2242–2251.

[14] W. Deng, L. Zheng, Q. Ye, G. Kang, Y. Yang, and J. Jiao, “Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 994–1003.

[15] Y. Li, L. Yuan, and N. Vasconcelos, “Bidirectional learning for domain adaptation of semantic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 6936–6945.

[16] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang, “CrDoCo: Pixel-level domain transfer with cross-domain consistency,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1791–1800.

[17] W. M. Kouv, L. J. Van Der Maaten, J. H. Krijthe, and M. Loog, “Feature-level domain adaptation,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 5943–5974, 2016.

[18] B. Sun, J. Feng, and K. Saenko, “Return of frustratingly easy domain adaptation,” in Proc. AAAI Conf. Artif. Intell., 2016, pp. 2058–2065.

[19] M. Long, Y. Cao, J. Wang, and M. Jordan, “Learning transferable features with deep adaptation networks,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 97–105.

[20] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” 2014, arXiv:1412.3474.

[21] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, “Moment matching for multi-source domain adaptation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 1406–1415.

[22] Z. Zhang, M. Wang, Y. Huang, and A. Nehorai, “Aligning infinite-dimensional covariance matrices in reproducing kernel hilbert spaces for domain adaptation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3437–3445.

[23] Y. Zou, X. Yang, Z. Yu, V. Kumar, and J. Kautz, “Joint disentangling and adaptation for cross-domain person re-identification,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 87–104.

[24] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, “VisDA: The visual domain adaptation challenge,” 2017, arXiv:1710.06924.

[25] Y. Wang, S. Liao, and L. Shao, “Surpassing real-world source training data: Random 3D characters for generalizable person re-identification,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 3422–3430.

[26] T. Zhang et al., “UnrealPerson: An adaptive pipeline towards costless person re-identification,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 11506–11515.

[27] Z. Tang et al., “PAMTIK: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 211–220.

[28] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2016, pp. 4340–4349.

[29] J. Xue, W. Mao, and L. Zheng, “Learning to simulate complex scenes for street scene segmentation,” IEEE Trans. Multimedia, vol. 24, pp. 1253–1265, 2021.

[30] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, “AI2-thor: An interactive 3D environment for visual AI,” 2017, arXiv:1712.05474.

[31] Y. Hou, L. Zheng, and S. Gould, “Multiview detection with feature perspective deep transformation,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 1–18.

[32] S. Xiang, Y. Fu, G. You, and T. Liu, “Unsupervised domain adaptation through synthesis for person re-identification,” in Proc. IEEE Int. Conf. Multimedia Expo, 2020, pp. 1–6.

[33] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, and A. G. Schwing, “SAIL-VOS: Semantic amodal instance level video object segmentation-a synthetic dataset and benchmark,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 3100–3110.

[34] Q. Wang, J. Gao, W. Lin, and Y. Yuan, “Learning from synthetic data for crowd counting in the wild,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 8198–8207.

[35] A.-D. Doan, A. M. Jawaid, T.-T. Do, and T.-J. Chin, “G2D: From GTA to data,” 2018, arXiv:1806.07381.

[36] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” 2017, arXiv:1711.03938.
Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, “CamStyle: A novel data

M. Deitke et al., “Robothor: An open simulation-to-real embodied ai

2504 IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 46, NO. 4, APRIL 2024

A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in

Y. Zhou and L. Shao, “Aware attentive multi-view inference for vehicle re-

E. Ristani, F. Solera, R. Zou, F. Cattaneo, and C. Tomasi, “Performance

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image

K. Simonyan and A. Zisserman, “Very deep convolutional networks for

T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in

K. Devaranjan, A. Kar, and S. Fidler, “Meta-sim2: Unsupervised learning of

D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler, “Neural

Y. Zhang, M. Hassan, H. Neumann, M. J. Black, and S. Tang, “Generating

M. Mozkir, A. Zhang, J. Pineau, and D. Meger, “Intervention design for

R. K. Jones et al., “Learning to generate programs for 3D shape structure

A. Kuznetsova et al., “The open images dataset V4,”

L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable

K. Simonyan and A. Zisserman, “Very deep convolutional networks for

J. Devaranjan, A. Kar, and S. Fidler, “Meta-sim2: Unsupervised learning of

K. Devaranjan, A. Kar, and S. Fidler, “Scene-structure learning for synthetic
data generation,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 715–733.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet:
A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput.
Vis. Pattern Recognit., 2009, pp. 248–255.

T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in
Proc. Eur. Conf. Comput. Vis., 2014, pp. 740–755.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image
recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016,
pp. 770–777.

K. Simonoyan and A. Zisserman, “Very deep convolutional networks for
large-scale image recognition,” 2014, arXiv:1409.1556.

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., 2015, pp. 1116–1124.

E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance
measures and a data set for multi-target, multi-camera tracking,” in
Proc. Eur. Conf. Comput. Vis. Workshop Benchmarking Multi-Target Tracking,
2016, pp. 17–35.

D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler, “Neural
parts: Learning expressive 3d shape abstractions with invertible neural
networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021,
pp. 3204–3215.

J. Devaranjan, A. Kar, and S. Fidler, “Meta-sim2: Unsupervised learning of
scene structure for synthetic data generation,” in Proc. Eur. Conf. Comput.
Vis., 2020, pp. 3204–3215.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet:
A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput.
Vis. Pattern Recognit., 2009, pp. 248–255.

T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in
Proc. Eur. Conf. Comput. Vis., 2014, pp. 740–755.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image
recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016,
pp. 770–777.

K. Simonoyan and A. Zisserman, “Very deep convolutional networks for
large-scale image recognition,” 2014, arXiv:1409.1556.

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., 2015, pp. 1116–1124.

E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance
measures and a data set for multi-target, multi-camera tracking,” in
Proc. Eur. Conf. Comput. Vis. Workshop Benchmarking Multi-Target Tracking,
2016, pp. 17–35.

D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler, “Neural
parts: Learning expressive 3d shape abstractions with invertible neural
networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021,
pp. 3204–3215.

J. Devaranjan, A. Kar, and S. Fidler, “Meta-sim2: Unsupervised learning of
scene structure for synthetic data generation,” in Proc. Eur. Conf. Comput.
Vis., 2020, pp. 3204–3215.
Yue Yao (Member, IEEE) received the bachelor’s degree in advanced computing from the Australian National University, in 2018. He is currently working toward the PhD degree with the Australian National University. He is a research officer with the School of Computing, Australian National University. His research interests include training set optimization, data synthesis and brain-computer interface.

Liang Zheng (Senior Member, IEEE) received the BE degree in life science from Tsinghua University, China, in 2010 and the PhD degree in electronic engineering from Tsinghua University, China, in 2015. He is an associate professor with the School of Computing, Australian National University. He was a postdoc researcher with the Center for Artificial Intelligence, University of Technology Sydney, Australia (2016 - 2018). His research interests include object re-identification, domain adaptation, deep learning and data synthesis.

Xiaodong Yang received the BS degree from Huazhong University of Science and Technology, China, in 2009, and the PhD degree from City University of New York, USA, in 2015. He is the head of machine learning with QCraft. Previously, he was a senior research scientist with NVIDIA Research. His research interests include autonomous driving, image and video understanding, activity and gesture recognition, facial analytics, object re-identification, deep generative modeling, etc.

Milind Napthade received the PhD degree in electrical engineering from the University of Illinois at Urbana-Champaign. He is the senior vice president in Capital One. Previously he led the technology and innovation strategy and engineering execution for NVIDIA’s Metropolis platform. He previously served as the program director in the Smarter City Services Group with the IBM T.J. Watson Research Center in Hawthorne, New York, and leader of the Smarter Sustainable Dubuque living lab.

Tom Gedeon (Senior Member, IEEE) received the BSc and PhD degrees from the University of Western Australia, and Grad Dip Management from UNSW. He is the Human-Centric Advancements Chair in AI and Head of the Human-Centric Advancements group in Artificial Intelligence with Curtin University. He is an international research professor with Obuda University, in Hungary. He is an honorary professor with the Australian National University, where he was formerly deputy dean and head of computer science. He is twice a former president of the Asia-Pacific Neural Network Assembly, and former President of the Computing Research and Education Association of Australasia. He is an associate editor of IEEE Transactions on Fuzzy Systems, and the INNS/Elsevier journal Neural Networks. His research interests are in responsive AI and responsible AI.