Optimizing the Training Set to Improve Model Specificity on Target Domains

Yue Yao

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

December 2023

Typeset in Palatino by TEX and LATEX 2ε.

Except where otherwise indicated, this thesis is my own original work.

Yue Yao
6 December 2023

Acknowledgements

Throughout my Ph.D. journey, I faced many challenges, including the Australian bushfire and the global COVID-19 pandemic. However, I am fortunate to have the unwavering support of my supervisors, friends, and family. I would like to express my sincere gratitude to the following individuals:

- Prof. Tom Gedeon, my primary supervisor and panel chair, who has been instrumental in guiding me since I started as a summer scholar in 2017. Without his help, I would not have entered the world of research.
- Dr. Liang Zheng, my co-supervisor, for his invaluable support, guidance, and dedication throughout my Ph.D. journey.
- other members of my supervisor panel, Dr. Sabrina Caldwell and Dr. Md Zakir Hossain, for their professional advice and guidance on my work. I am also thankful for Dr. Richard Jones' help with my work.
- my friends at ANU, including Jing, Tianyu, Yuchen, Yuchi, Heming, Yuhang, Zhenyue, Yang, Xiaoxiao, Weijian, Weijie, Yunzhong, Ruitao, Ruyi, *etc*. I am grateful for our insightful research discussions and delicious meals together. Special thanks to my badminton friends for the wonderful time we had playing sports.
- my unparalleled collaborators, Dr. Milind Napthade and Dr. Tang Zheng, for hosting my internship at NVIDIA. I am also thankful for Dr. Xiaodong Yang's constructive feedback on my papers.
- my parents and my fiancée Zhengyang, whose love, support, encouragement, and understanding have been critical to my Ph.D.

This research is also supported in part by the ARC Discovery Project (DP210102801), the ARC Discovery Early Career Researcher Award (DE200101283), and Oracle Cloud credits, and related resources provided by Oracle for Research.

Publications

The research conducted and presented in this thesis herein is based on the publications listed below.

1. **Yue Yao**, Huan Lei, Tom Gedeon and Liang Zheng, *A Large-scale Search for Object Re-identification Training Data*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2023. Chapter 3.

2. **Yue Yao**, Liang Zheng, Xiaodong Yang, Milind Naphade and Tom Gedeon, *Attribute Descent: Simulating Object-Centric Datasets on the Content Level and Beyond*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 2023. Chapter 4.

3. **Yue Yao**, Liang Zheng, Xiaodong Yang, Milind Naphade and Tom Gedeon, *Simulating Content Consistent Vehicle Datasets with Attribute Descent*, European Conference on Computer Vision (ECCV) 2020. Chapter 4.

4. **Yue Yao**, Jo Plested and Tom Gedeon, *Information-preserving Feature Filter for Short-term EEG signals*, Neurocomputing 2020. Chapter 5.

5. **Yue Yao**, Jo Plested, Tom Gedeon, Yuchi Liu and Zhengjie Wang, *Improved Techniques for Building EEG Feature Filters*, International Joint Conference on Neural Networks (IJCNN) 2019. Chapter 5.

6. **Yue Yao***, Xinyu Tian*, Zheng Tang, Sujit Biswas, Huan Lei, Tom Gedeon and Liang Zheng, *Training with Product Digital Twins for AutoRetail Checkout*, Arxiv Preprint 2023, * denotes equal contribution. Chapter 6.

Besides, I have worked on the following papers that are less related to the thesis.

1. **Yue Yao**, Tianyu Wang, Heming Du, Liang Zheng and Tom Gedeon, *Spotting Visual Keywords from Temporal Sliding Windows*, International Conference on Multimodal Interaction (ICMI) 2019.

2. **Yue Yao**, Jo Plested and Tom Gedeon, *Deep Feature Learning and Visualization for EEG Recording Using Autoencoders*, International Conference on Neural Information Processing (ICONIP) 2018. **Nominated for Best Student Paper award**.

3. **Yue Yao**, Jo Plested and Tom Gedeon, *A Feature Filter for EEG Using Cycle-GAN Structure*, International Conference on Neural Information Processing (ICONIP) 2018.

4. Xiaoxiao Sun, **Yue Yao**, Shengjin Wang, Hongdong Li and Liang Zheng, *Alice Benchmarks: Connecting Real World Object Re-Identification with the Synthetic*, Arxiv Preprint 2023.

5. Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, **Yue Yao**, Liang Zheng, Mohammed Shaiqur Rahman, Archana Venkatachalapathy, Anuj Sharma, Qi Feng, Vitaly Ablavsky, Stan Sclaroff, Pranamesh Chakraborty, Alice Li, Shangru Li and Rama Chellappa, *The 6th ai city challenge*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2022.

6. Yuhao Zhang, **Yue Yao**, Md Zakir Hossain, Shafin Rahman and Tom Gedeon, *EEG Feature Significance Analysis*. International Conference on Neural Information Processing (ICONIP) 2021.

7. Milind Naphade, Shuo Wang, David C Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, **Yue Yao**, Liang Zheng, Pranamesh Chakraborty, Christian E Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky and Stan Sclaroff, *The 5th ai city challenge*, IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2021.

8. Xuyang Shen, Jo Plested, **Yue Yao** and Tom Gedeon, *Pairwise-GAN: Pose-Based View Synthesis Through Pair-Wise Training*, International Conference on Neural Information Processing (ICONIP) 2020.

9. Shiya Liu, **Yue Yao**, Chaoyue Xing and Tom Gedeon, *Disguising Personal Identity Information in EEG Signals*, International Conference on Neural Information Processing (ICONIP) 2020.

10. Yuchi Liu, **Yue Yao**, Zhengjie Wang, Jo Plested and Tom Gedeon, *Generalized Alignment for Multimodal Physiological Signal Learning*, International Joint Conference on Neural Networks (IJCNN) 2019.

Abstract

Over the last few decades, researchers have worked to improve deep learning models' specificity on a target domain, striving to achieve human-level performance. To reach such a level, the development of deep learning has relied on the joint confluence of the model (*i.e.*, algorithms) and data. Noticeably, the optimization of data has received relatively little attention in comparison to the optimization of the model. This thesis aims to strengthen deep learning models' specificity from a data-centric perspective. We highlight that the training data, *i.e.*, the data which deep learning models learned from, has significant potential for improvement. Specifically, we analyzed the training set quality, and designed algorithms to perform training set optimization, under various scenarios.

We first aim to understand the training set quality, which is critical to the success of the resulting model trained on it, and forms the foundation of this thesis. In this thesis, we verify that data diversity and domain gap are two pivotal aspects describing training data quality. The former encodes the extent to which training samples differ, and the latter characterizes the distribution difference between training and testing. We argue that the two factors are intrinsically correlated and should be jointly considered. We formulate training data quality as a function of both data diversity and domain gap, and quantitatively analyze it on 872 different training sets created by geometric image transformations and neural rendering. We verify that a joint objective of high diversity and low domain gap characterize high-quality training data, and apply this idea in composing better training sets.

We then focus on an application called training set search, in which we aim to construct a competent training set under certain scenarios. We first consider a scenario where we have access to the target domain, but cannot afford on-the-fly training data annotation, and instead would like to construct an alternative training set from a large-scale data pool such that a competitive model can be obtained. We propose a search and pruning (SnP) solution to this training data search problem, tailored to object re-identification (re-ID), an application aiming to match the same object captured by different cameras. The SnP solution provides us with training sets 80% smaller than the source pool while achieving a similar or even higher re-ID accuracy.

We further consider a training set optimization scenario where we have access to the target domain, and instead would like to use synthetic data to construct an alternative training set. To address the content-level misalignment between synthetic and real, we propose an attribute descent approach that automatically optimizes engine attributes to enable synthetic data to approximate real-world data. Extensive experiments on image classification and object re-ID confirm that adapted synthetic data with attribute descent can be effectively used in three scenarios: training with syn-

thetic data only, training data augmentation and numerically understanding dataset content. The scenario of training with synthetic data only relates to data privacy and security at the corporate level, as the expensive and confidential private data can be protected while a model is trained on synthetic data only.

Apart from studying training set quality on training a high accuracy model, we also consider building a privacy-preserved training set which leads to privacy-preserving models for people data. We studied the feature fusion problem in data, and proposed a new strategy to filter out unwanted features from data based on our feature extractor, tailored to the application of brain-computer interface, which its input signal electroencephalography (EEG) is known as feature fused and rich in all kinds of information from our brain. Our experimental results on an alcoholism dataset show that our novel model can filter out over 90% of alcoholism information on average from EEG signals, with an average of only 4.2% useful feature accuracy lost, showing effectiveness for our proposed task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

At its core, constructing a task-specific deep learning (DL) system involves two pillars: the model (*i.e.*, algorithm and code) and data it trained on. To explain, the DL system solves problems by using a model to learn prototypical features from vast amounts of data. Given such a pipeline, there are two main approaches to improving the performance of DL systems: model-centric and data-centric. Shown in Fig. 1.1 left, in model-centric DL, developers of a DL system focus on upgrading the model (algorithm/code) while keeping the amount and type of data collected fixed. This approach involves tweaking the model's architecture and optimizing its hyperparameters, *etc*. [4, 5]. Conversely, in data-centric DL, shown in Fig. 1.1 right, practitioners focus on keeping the model fixed, but working on improving the quality of data. This approach involves identifying and adapting bias in the target data [6], augmenting the data with new samples or features [7], and collecting additional data (*e.g.*, synthetic data) [8] to increase the diversity and coverage of the dataset.

Model-centric DL focuses on iterating model building to develop strong DL models. This approach involves designing and fine-tuning neural network architectures to achieve the desired level of accuracy. Such a process usually requires the usage of a benchmark, which means a fixed, standard dataset for evaluating deep learning models' performance and selecting competent models for deployment [9]. For exam-

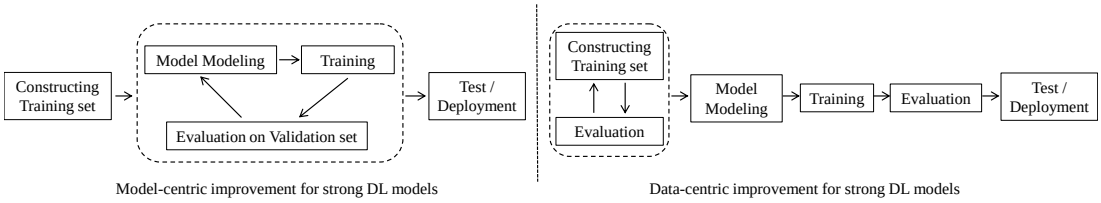

Figure 1.1: The difference between model-centric DL and data-centric DL. The model-centric DL aims to have strong DL models by iterating model building. In comparison, the data-centric DL aims to have strong DL models by iterating training set construction.

ple, ImageNet [10] is one of the most famous benchmarks for evaluating and selecting image classification models. The usage of benchmarks successfully results in leading deep learning models. For example, the usage of the ImageNet benchmark has successfully introduced computer vision models including VGG [11], GoogleNet [12], ResNet [13], *etc*. However, despite its dominance over the past decades, model-centric AI has been recently criticized for problems like the requirement of a large amount of computational resources, difficulty in hyperparameter tuning, weak generalization ability, saturation in existing benchmarks and bias [9].

In contrast to model-centric deep learning (DL), which focuses on improving the architecture of DL models, data-centric DL places greater emphasis on the construction of a high-quality training set through iterative selection and preprocessing of data. As depicted in Fig 1.1, this approach involves selecting or generating representative data to create a training set that accurately reflects the problem being addressed. Similar to the benchmarks set to evaluate deep learning models, we establish benchmarks for evaluating training set quality, *i.e*., evaluating the ability to train competent deep learning models. Specifically, we imitate the model benchmark setting, where we have access to a target validation set and test set.

In early papers, benchmarking training sets typically require multiple iterations of real training processes to assess the model's accuracy on the validation set, which can be time-consuming and computationally expensive. For instance, Cubuk *et al*. employed real training processes to enhance the data augmentation policy on ImageNet [14]. But such a process requires 15,000 GPU hours with the NVIDIA P100 GPU, showing that using the validation accuracy for constructing good training sets is very time-consuming.

In this thesis, we are among the first to overcome the need for real training processes by developing a method for identifying and constructing a good training set. We examine the *essential composition factors* of a high-quality training set and apply our findings to a novel task known as *training set search*. Specifically, we have the following objectives.

1.2 Objectives

The first objective of this work is to provide evaluation metrics to evaluate the quality of the training set, a factor that is crucial to the effectiveness of the resulting models and is the cornerstone of this thesis. The conventional approach involves undertaking actual training to gauge quality, but this method is time-consuming, making quality assessment challenging. In this thesis, we aim to evaluate the training set quality prior to actual training. We emphasize that data diversity and the domain gap are two critical factors that define training data. Data diversity pertains to the degree of variance among training samples, whereas the domain gap signifies the disparity in distribution between training and testing data. Insufficient diversity can lead to model overfitting, and data augmentation is commonly employed to mitigate this issue. A significant domain gap between training and testing can hinder learning. In

our study, we found target domain usually exhibits data bias like distinct modes in filming scenes, if the training set does not contain these target modes (thus a large domain gap), the model performance would be compromised. Current literature addresses this problem with techniques such as pixel-level alignment (image translation) and content-level alignment (image content manipulation). There is a dearth of research that concurrently considers diversity and domain gaps in understanding training set quality. In this research, we posit that these two factors are inherently related and should be evaluated collectively. We define training data quality as a function of both data diversity and the domain gap, and conduct a quantitative analysis on 872 diverse training sets, formulated by two distinct methods: geometric image transformations and neural rendering. Through comprehensive experiments, we offer profound insights into measuring and ensuring the quality of training data. We confirm that an integrated goal of high diversity and minimal domain gap signifies high-quality training data and employ this concept to create improved training sets.

Our next area of focus is an application we have named training set search, which seeks to build an efficient training set suitable for a specific target. Initially, we look at a situation where we have access to the target domain but lack the resources for immediate training data annotation. Instead, we aim to create a substitute training set from a substantial data pool to generate a competitive model. We introduce a search and pruning (SnP) strategy to address this training data search issue, specifically for object re-identification (re-ID), a task designed to match the same object captured by various cameras. More precisely, the search phase identifies and combines clusters of source identities displaying similar distributions (*i.e.*, a small domain gap) with the target domain. The subsequent stage, within a specified budget, selects identities and their images from the results of the first stage to manage the size of the resulting training set for efficient training. These two stages produce training sets that are 80% smaller than the source pool, while maintaining or even enhancing re-ID accuracy. These training sets also outperform several existing search methods such as random sampling and greedy sampling, given the same budget for training data size. If the budget constraint is lifted, training sets generated from the first stage alone can deliver even greater re-ID accuracy.

We also explore a training set search scenario in which we have access to the target domain and wish to use synthetic data to build an alternate training set. Two distinct levels of domain gaps are present between synthetic and real data, namely, content level and appearance level. While the appearance level pertains to the style of appearance, the content level results from differences in aspects like camera viewpoint, object positioning, and lighting conditions. The content-level discrepancy, unlike the widely researched appearance-level gap, has not received much attention. To tackle this content-level misalignment, we suggest an attribute descent method that automatically refines engine attributes, allowing synthetic data to closely mirror real-world data. We test our approach on object-centric tasks where an object occupies a significant portion of an image. In such tasks, the search space is fairly limited, and optimizing each attribute provides distinct supervisory signals. We assemble a new synthetic asset named VehicleX and reconfigure and repurpose existing synthetic as-

sets ObjectX and PersonX. Comprehensive experiments on image classification and object re-identification affirm that adapted synthetic data can be effectively applied in three scenarios: solely training with synthetic data, training data augmentation, and numerically understanding dataset content. The ability to train with synthetic data only is important for data privacy and security for organizations, as the expensive or confidential private data can be protected while a model is trained on synthetic data only.

In addition to exploring the quality of the training set for generating high-accuracy models, we also examine the creation of privacy-preserving training sets that result in privacy-conscious models. We delved into the issue of feature fusion in data and introduced a new approach to eliminate undesired features from data utilizing our feature extractor. This is designed specifically for the brain-computer interface, with its input signal, electroencephalography (EEG), known for its feature fusion and the rich array of information sourced from our brain. The process of filtering out signals associated with one aspect of the EEG signal while keeping another mirrors our ability to focus on a single voice in a crowded party, a phenomenon known in the machine learning field as the cocktail party problem. Our feature filter is an end-to-end framework that is trained to transform EEG signals replete with undesired features directly into EEG signals devoid of those features. The experimental outcomes using an alcoholism dataset demonstrate that our innovative model can filter out over 90% of alcoholism information on average from EEG signals, with an average loss of only 4.2% in useful feature accuracy, thereby proving its efficacy for our proposed task.

1.3 Contributions

Targeting these objectives outlined above, this thesis presents new contributions that are itemized as follows:

- We have analyzed the intent composition of a good training set. We verify that a joint objective of high diversity and low domain gap characterize high-quality training data, and apply this idea in composing better training sets.
- We propose a search and pruning (SnP) solution to the training data search problem in the large source pool. The two steps provide us with training sets 80% smaller than the source pool while achieving a similar or even higher task accuracy.
- We propose an attribute descent approach that helps to use synthetic data to construct training sets for model training. The proposed approach automatically optimizes engine attributes to enable synthetic data to approximate real-world data.
- We studied the feature fusion problem in data, and proposed a feature filter to filter out unwanted features, for the purpose of building privacy-preserving training data.

1.4 Thesis Outline

This dissertation is organized as follows:

- In this chapter, we have briefly introduced our motivations, objectives as well as significant contributions.
- Chapter 2 establishes essential foundations for the training set search by analyzing how to compose a good quality training set.
- Chapter 3 describes training data search from a source pool.
- Chapter 4 presents training data optimization from synthetic to real.
- Chapter 5 shows approaches that can be used to filter out unwanted features in the training set.
- Finally, in chapter 6, we conclude our thesis and discuss several future directions for data-centric optimization.

Chapter 2

Training Set Quality Analysis

2.1 Introduction

As a pillar in machine learning, a high-quality training set helps to achieve good accuracy on the test set. According to textbook definitions, an ideal training set should have exactly the same distribution as the test set, which is very difficult to achieve in real-world scenarios. In practice, such an assumption is rarely satisfied, and we often find limited diversity in the training data; or a large domain gap between training and testing data.

Data diversity reflects how training samples differ from each other. Limited training data diversity can cause overfitting or performance degradation of models. In order to improve data diversity, existing work [14, 15, 16] usually apply data augmentation on the training data (Fig. 2.1 (B)), *e.g.*, flipping, cropping, brightness adjustments, and cutout [17, 18]. This leads to a higher diversity of training data, which increases training data quality in the sense that it results in more robust models.

Domain gap characterizes the distribution difference between training and testing. A large domain gap indicates severe distribution drift and creates difficulties for learning. Many existing works in domain adaptation try to generate training sets with reduced domain gaps. Some achieve this by neural renderings with image translation [19] so that the training images share similar artistic styles to the testing images (Fig. 2.1 (C)). Others adopt content-level adaptation by directly manipulating the image contents such as object class, identity, characteristics, and scene layout [20, 21, 22, 23], and create a training set that has similar contents as the test set. Both pixel-level and content-level help to bridge the domain gap between training and testing, and create higher-quality training data that allows for better performance of the model.

Both data diversity and domain gap characterize the training data quality in that different levels lead to different model accuracies on the test set. In the literature, very few works consider both of them when mentioning training data quality. In this thesis, we jointly consider these two concepts for training data quality assessment, and find that diversity and domain gap are correlated. We quantitatively analyze training data quality as a function of both data diversity and domain gaps based on our proposed measurements. Specifically, we measure the training data quality as the test accuracy.

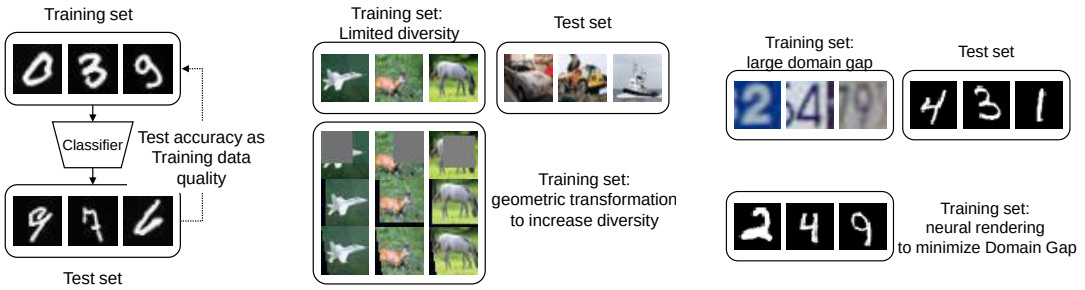

Figure 2.1: The definition of training data quality and operations that influence it. (Left): We evaluate the training data quality as the test accuracy from a trained model. Rather than the ideal situation where the training data follows the same distribution as the testing data, in real-world applications, oftentimes the training data suffers from (Middle): limited diversity or (Right): a large domain gap between training and testing. Many methods are proposed to address these problems, *e.g.*, geometric image transformations to increase diversity, and neural rendering to bridge the domain gap.

For diversity and domain gap, we use both unsupervised measurements that only focus on the image distributions, and supervised measurements that jointly consider both images and labels.

To enable our analysis, we design and conduct extensive experiments to showcase the influence of domain gaps and data diversity on the training data quality. For these experiments, we create 872 different training sets with 53 million images on 6 different settings using two types of methods: geometric transformations and neural rendering. We train models on these 872 datasets for quality evaluation, and consider each dataset with its quality, diversity measurement, and domain gap measurement as a "data point". We then scatter plot these "data points", and examine the plotted results for further study. Based on these experiments, we have some interesting findings. To name a few,

- Diversity and domain gap are correlated. A change in one of them often affects the other.
- Neither diversity nor domain gap can determine training data quality by itself.
- Diversity and domain gap together measure the quality of a training set. A training set with high diversity and low domain gap performs better on testing.
- There often exists a trade-off between diversity and domain gap. An increase in diversity is often accompanied by an unwanted increase in domain gap.

Our findings not only provide insights into the scientific problem of training set assessments, but also have real-world applications such as searching for a training dataset with the highest performance. To be shown in the following chapters, we consider creating a substitute training set from a substantial data pool, or generating alternative training set from synthetic data, to generate competitive models. In the coming chapters, we show that with an objective function that describes the domain gap only, the searched training sets are well enough to result in competitive models.

2.2 Related Work

Distribution Analysis. Recently, generative adversarial networks (GANs) achieve outstanding results in generating realistic images [24, 25]. The objective of GANs is to generate images that follow real-data distribution. In general, distribution similarity and diversity are used to measure quality. For example, inception score (IS) [26], kernel inception distance (KID) [27] and Fréchet inception distance (FID) [28] are usually used to measure *distribution similarity*. and structural similarity index (SSIM) [29], and learned perceptual image patch similarity (LPIPS) [30] is used to measure diversity. We extend these metrics to measure the quality of a training set. Different from distribution analysis on GANs which only consider the quality of data, the quality of a training set should be determined by the joint distribution of data and its labels.

Data augmentation is widely used in training a deep neural network. The aim is to enlarge the training set using various transformations, such as random crop, random erasing, rotation, flipping, and so on [7, 18], thereby improving model generalization ability. A large number of data augmentation policies are proposed in the past, which results in many combinations of methods. [14] shows that these policies can also be learned automatically. Earlier works mainly focus on increasing diversity and then increasing the training difficulty as shown in Fig. 2.1. But as some transformations bring changes that are unlikely to happen in the real world. In those cases, it will inevitably change the original data distributions and thus enlarge the domain gap. We also use data augmentation as a part of the training set quality analysis from a joint perspective of domain gap and diversity, and study how these two parameters influence the quality of a training set.

Domain adaptation has attracted more attention in the recent progress of deep learning. Existing approaches attempt to eliminate the domain gap between the source and target distributions [31, 32, 33]; also, multiple matrices are introduced to measure such domain gaps. For example, at the feature level, [34] and [32] utilize the maximum mean discrepancy (MMD) [35] to learn robust features across domains. At the style level, an end-to-end mapping with autoencoder structure is often used to transfer image style [19, 36]. Recent work also aims to learn a mapping from the content level using graphic engines [21, 20, 22, 37]. As shown in Fig. 2.1, existing work on domain adaptation mainly focuses on reducing the domain gap. Our work indicates reducing the domain gap is usually not enough and increasing diversity is also necessary.

Dataset bias and assessment. People collect datasets with the purpose, that they can reflect real-world distributions with no known bias. However, such a purpose is hard to meet and the reality is that different datasets have different biases. For example, Torralba *et al*. study dataset bias by training a classifier to determine which dataset a given image comes from, and get 39% accuracy which is significantly above the random chance of 8% [38]. Similar bias has been found in other applications like action recognition [39] and object re-identification (re-ID) [40, 41]. This thesis acknowledges that datasets (*i.e.*, both training and test set) inevitably have bias and define a new framework. We have a reference test set that defines cases in test scenarios including preference and bias. With this and fixed neural network, we study how we can

formulate a training set with the best quality.

2.3 Problem Formulation

Without loss of generality, we consider a machine learning task that learns a model $m_\theta(\cdot)$ with parameters θ to estimate labels for images $\boldsymbol{x}$. We use a loss function $\mathcal{L}(\cdot,\cdot)$ between estimated labels $m_\theta(\boldsymbol{x})$ and the ground truth labels y to optimize this model $m_\theta(\cdot)$,

$$\theta^*_{D_{\text{train}}} = \arg\min_\theta \sum_{\langle \boldsymbol{x}, y\rangle \in D_{\text{train}}} \mathcal{L}\left(m_\theta(\boldsymbol{x}), y\right), \tag{2.1}$$

where $D_{\text{train}} = \langle X_{\text{train}}, Y_{\text{train}}\rangle$ denotes the training set (X_{train} denotes the set of training images and Y_{train} denotes the set of corresponding ground truth labels), and $\theta^*_{D_{\text{train}}}$ denotes the best parameters for the model learned from the training set. We further denote the learned model as $m_{D_{\text{train}}}(\cdot)$.

In this thesis, we aim to understand the quality of a training set D_{train} for machine learning tasks. A training set of high quality should be able to train neural network models that can perform well on the test set. We use the test set $D_{\text{test}} = \langle X_{\text{test}}, Y_{\text{test}}\rangle$ accuracy of the trained model to represent the quality of that training set. We understand a training set from two perspective: data diversity and domain gap. We formulate the training set quality $\mathcal{Q}(D_{\text{train}})$ as a function of diversity and domain gap,

$$\begin{aligned} & \mathcal{Q}(D_{\text{train}}) \\ \rightarrow\ & \mathcal{Q}\left(div(D_{\text{train}}), gap(D_{\text{train}}, D_{\text{val}})\right) \\ \rightarrow\ & acc\left(m_{D_{\text{train}}}, D_{\text{test}}\right), \end{aligned} \tag{2.2}$$

where $div(\cdot)$ and $gap(\cdot,\cdot)$ denote the measurements of diversity and domain gap, respectively. To avoid using the test set other than evaluation, we use the validation set $D_{\text{val}} = \langle X_{\text{val}}, Y_{\text{val}}\rangle$ for diversity and domain gap measurements. Also, $acc\left(m_{D_{\text{train}}}, D_{\text{test}}\right)$ denotes the test set D_{test} accuracy of the model $m_{D_{\text{train}}}$.

2.3.1 Pipeline

In the search for the training set quality as a function of diversity and domain gap (Eq. 2.2), it is vital that we have proper measurements of these two concepts. Existing measurements are usually *unsupervised*, which means that data diversity and domain gap are estimated based on images only, and can be formulated as functions of the images $div(X_{\text{train}})$ and $gap(X_{\text{train}}, X_{\text{val}})$.

We aim to understand training data quality for machine learning, and machine learning usually relies on both images and labels. As such, measurements of diversity and domain gap should be formulated as functions of both images and their labels, $div(D_{\text{train}}) = div(\langle X_{\text{train}}, Y_{\text{train}}\rangle)$, $gap(D_{\text{train}}, D_{\text{val}}) = gap(\langle X_{\text{train}}, Y_{\text{train}}\rangle, \langle X_{\text{val}}, Y_{\text{val}}\rangle)$,

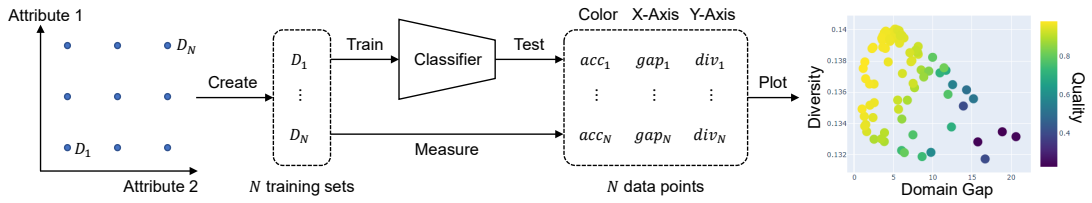

Figure 2.2: The proposed method for investigating training set quality. To analyze the training data quality as a function of both diversity and domain gap, first, through controllable attributes, we create N different training sets $\{D_n\}, n \in \{1, ..., N\}$. Then, we can measure the diversity $\{div_n\}$ and domain gap $\{gap_n\}$ of these datasets, and get their quality as the evaluation results from trained models $\{acc_n\}$. Finally, we can plot figures to show the training data quality as a function of diversity and domain gap, by interpreting diversity and domain gap as the x-axis and the y-axis, respectively.

rather than functions of images only. In what follows, we refer to these as *supervised* measurements, which means not only images but also their labels are considered. In the next sections, we also investigate *unsupervised* measurements of data diversity and domain gap to enable our investigation of training data quality $\mathcal{Q}(D_{\text{train}})$.

2.3.2 Measurement of Diversity

Diversity encodes the averaged difference between training samples. However, to the best of our knowledge, there is no evaluation protocol for diagnosing the *overall* diversity of a dataset. Existing work has used statistics over image samples like Structural SIMilarity (SSIM) [29] and Learned Perceptual Image Patch Similarity (LPIPS) [30] to define the difference between two images. Both SSIM and LPIPS are *unsupervised*, as their calculation does not require labels. To accurately measure the overall difference of samples in the feature space, we define the matrix $\mathcal{W}$ as,

$$\mathcal{W} = \begin{bmatrix} f(X_1, X_1) & f(X_1, X_2) & \cdots & f(X_1, X_N) \\ f(X_2, X_1) & f(X_2, X_2) & \cdots & f(X_2, X_N) \\ \vdots & \vdots & & \vdots \\ f(X_N, X_1) & f(X_N, X_2) & \cdots & f(X_N, X_N) \end{bmatrix}, \tag{2.3}$$

where $f(\cdot)$ denotes the difference between two images in the feature space, *e.g.*, SSIM [29] and LPIPS [30]. Given $\mathcal{W}$, we define *unsupervised* metric to denote the overall diversity:

$$div(D_{\text{train}}) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} f(X_i, X_j)}{N^2 - N}, i \neq j. \tag{2.4}$$

In this chapter, we propose a *supervised* diversity measurement using the loss function $\mathcal{L}(\cdot, \cdot)$. Since higher diversity increases the difficulties of fitting a model (and pre-

vents overfitting), we can use the training difficulty to represent its diversity. Given a model with fixed architecture, higher difficulties of the training set can be indicated by higher overall loss value given by a trained model. We thus define the diversity measurement as,

$$div\left(D_{\text{train}}\right) = \sum_{\langle \boldsymbol{x}, y\rangle \in D_{\text{train}}} \mathcal{L}\left(m_{D_{\text{train}}}\left(\boldsymbol{x}\right), y\right). \tag{2.5}$$

This measurement comes from a dataset point-of-view as its calculation uses both images and labels. In Fig. 2.3, we experimentally show that increased diversity caused by a higher level of data augmentation leads to an increase of the proposed diversity measurement in Eq. 2.5.

2.3.3 Measurement of Domain Gap

Domain gap refers to the distribution difference between the training and test sets. Currently, unsupervised distribution measurements like Fréchet distance (FD) [28] and kernel distance (KD) [42] are widely adopted for measuring the domain gap. Similar to SSIM and LPIPS mentioned above for diversity measurement, FD and KD are also *unsupervised* measurements, as their calculations only use images. For example, the FD is defined as

$$\begin{aligned} \text{FD}(D_{train}, D_{val}) = \|\boldsymbol{\mu_t} - \boldsymbol{\mu_v}\|_2^2 + \\ Tr(\boldsymbol{\Sigma_t} + \boldsymbol{\Sigma_r} - 2(\boldsymbol{\Sigma_t}\boldsymbol{\Sigma_r})^{\frac{1}{2}}), \end{aligned} \tag{2.6}$$

where $\boldsymbol{\mu}_t \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_t \in \mathbb{R}^{d\times d}$ denote the mean and covariance matrix of the features extracted from a pre-trained model, respectively.

For measuring the domain gap in a supervised manner, we use the model accuracy difference (affinity) between the training set and the validation set to represent the domain gap, which is proposed by Lopes *et al*. [43]. Given a model trained in one domain, when tested in another domain, the larger the domain gap, the more difficult it is for the model to generalize well. As such, we measure the domain gap as,

$$gap\left(D_{\text{train}}, D_{\text{val}}\right) = acc\left(m_{D_{\text{val}}}, D_{\text{val}}\right) - acc\left(m_{D_{\text{val}}}, D_{\text{train}}\right), \tag{2.7}$$

where $m_{D_{\text{val}}}$ denotes a model trained on the validation set. This approach has a low calculation cost as only one model needs to be trained for the measurement. It is also noteworthy that the model trained on the validation set $m_{D_{\text{val}}}(\cdot)$ is used and only used for the calculation of the domain gap measurement, and we still train all other models normally on the training set to evaluate the training data.

As the proposed domain gap measurement is calculated using labels, it is also *supervised*. In Fig. 2.3, we show that the proposed domain gap measurement increases as we deliberately increase the domain gap by rotating the training images. In Section 2.5 Fig. 2.5, we further show that neither of the proposed measurements alone determines the training data quality.

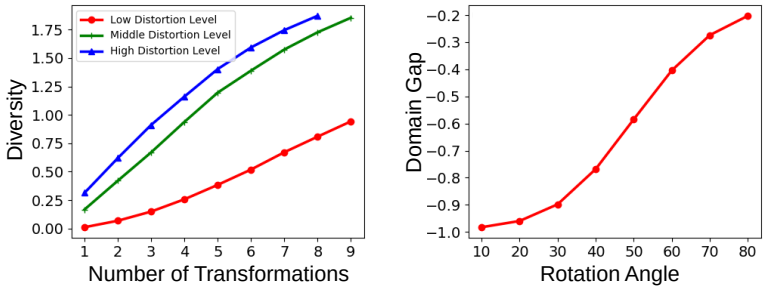

Figure 2.3: Geometric image transformations influence both diversity and domain gap. **Left:** As more transformations are applied to the training data or the distortion levels (degree of the transformations) increase, the diversity increases, and the proposed measurement of diversity (y-axis) also increases. **Right:** In the rotating and translating MNIST experiment, as we increase the rotation angle of the training images, we create a larger domain gap between training and testing, and we witness an increase in the proposed measurement of the domain gap (y-axis).

2.4 Experiment

In order to understand the training data quality, we design and conduct extensive experiments. In the following sections, we first introduce how we design these experiments attempting to show that the training data quality is a function of both data diversity and domain gap (Eq. 2.2). Then, we detail the methods we adopted in these experiments.

2.4.1 Experiment Design

In our experimental study, to understand training data quality, we first generate different training sets. As shown in Fig. 2.2, we use controllable attributes to generate N different training sets $\{D_n\}, n \in \{1, ..., N\}$. These N training sets have diversity measurements of $\{div_n\} = \{div(D_n)\}$ and domain gap measurements of $\{gap_n\} = \{gap(D_n, D_{\text{val}})\}$. Next, we train models on these training sets and evaluate the training set qualities using these trained models $\{m_{D_n}(\cdot)\}$. Once we have N training sets and their corresponding quality evaluations $\{acc_n\} = \{acc(m_{D_n}, D_{\text{val}})\}$, we represent each created dataset as a "data point" using a three-tuple $\langle acc_n, gap_n, div_n \rangle$. As shown in Fig. 2.2, we then interpret domain gap and diversity as the x-axis and the y-axis, respectively, so that we can show the data quality (color) as a function of both of them.

2.4.2 Training Sets Creation

In order to create different training sets in a controllable manner, we first explore a toy experiment on rotating and translating MNIST. We explore toy examples of rotating

and translating MNIST and further use two methods: geometric image transformation, and neural rendering.

Rotating and translating MNIST is our toy preliminary experiment, which modifies controllable attributes in a graphic engine or other rendering methods to directly change the appearance of image contents [21, 20, 22]. Different combinations of attribute changes lead to various derived datasets.

Following the setting from Kar *et al*. [21], given MNIST images that are either rotated or rotated and translated as test sets, we create training sets by directly manipulating the rotation angles and the translation biases. Specifically, we model the rotation angle and the translation bias as Gaussian distributions, and control the mean and variance to generate different training sets. For both settings on MNIST rotation and MNIST rotation+translation, we create 200 training sets. Similar to the neural rendering experiments, we also use the LeNet [44] classifier and train it with AdaDelta optimizer [45] for 15 epochs.

Geometric image transformation is commonly used in data augmentation and is widely used to increase dataset diversity using methods like flipping, cropping, brightness adjustment, and cutout [17, 18]. Different numbers of applied transformations and different distortion levels will produce derived datasets with different diversity levels (Fig. 2.3).

We use geometric image transformation to create different versions of CIFAR-10 [46] and SVHN [47] datasets. Specifically, we use 14 types of data augmentation methods following randAugmentation [48]. To generate different training sets in a controllable manner, we select different numbers of transformations and different distortion levels (similar to the experiment in Fig. 2.3). For both CIFAR-10 and SVHN, for each setting, we create 196 different training sets using different data augmentations. On the created datasets, we train a WideResNet [49] classifier using SGD optimizer for 200 epochs.

Neural rendering uses neural networks to transfer image style to another domain. Different losses and hyperparameters result in various transformation results and thus create different derived datasets.

We use the neural rendering method CyCADA [19] to create different training sets for the domain adaptation task MNIST→USPS and SVHN→MNIST. CyCADA trains a generator to transfer training data to the testing style. We use a generator trained for different numbers of epochs to create different training sets. We create 40 different training sets for both MNIST→USPS and SVHN→MNIST. On the created datasets, we train a LeNet [44] classifier using AdaDelta optimizer [45] for 15 epochs.

These three methods are usually linked with either diversity or domain gap (Fig. 2.3), and we believe these methods all bring changes to both of them. In our experiment, we use these methods separately to create modified training sets $\{D_n\}$.

Summary. In total, we experiment under 6 settings (2 for each training set creation method) and generated 872 different datasets, which translates into a total of 53 million training images and labels. In each setting, each training set represents a data point, and we scatter these data points on a xy coordinate system with domain gap as x-axis value and diversity as y-axis value.

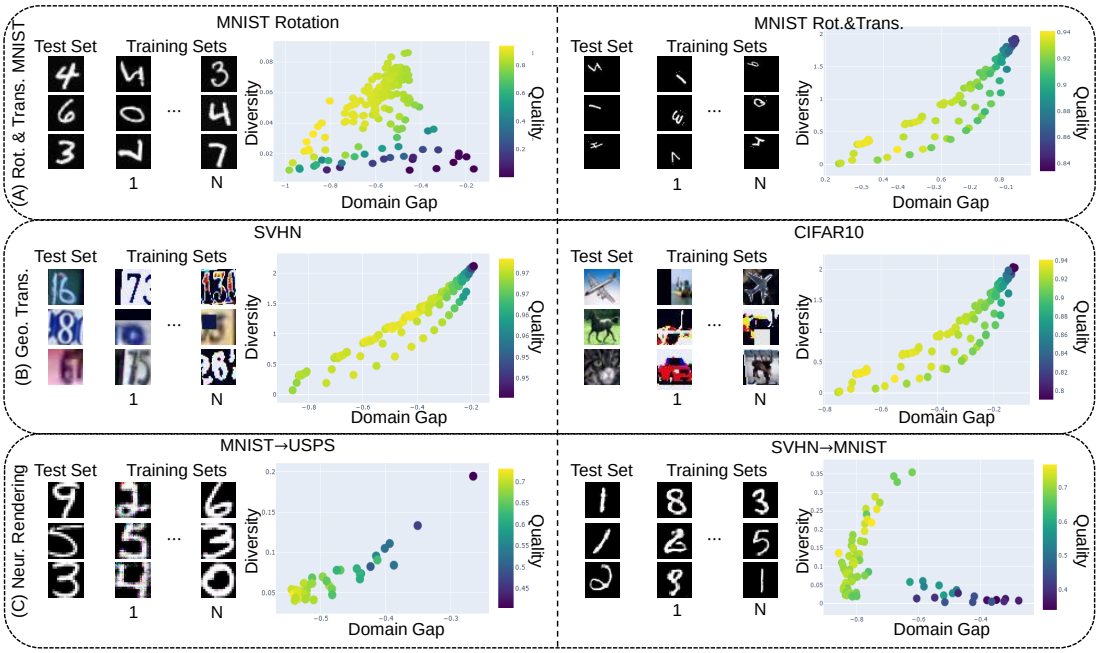

Figure 2.4: Relationship between domain gap, diversity and training set quality on experiments (A) rotating and translating MNIST, (B) geometric image transformation and (C) neural rendering. Each point in the scatter plot represents a training set. The color of each point shows the training set quality (*i.e.*, test accuracy). These six experiments show that a relatively low domain gap and high diversity are desired.

2.5 Results and Findings

We show our results on the influence of diversity and domain gap on the training data quality in Fig. 2.4 and Fig. 2.5 using the *supervised* measurements. We also examine the training data quality using *unsupervised* measurements of domain gap and diversity. In some scenarios, we have similar findings using *unsupervised* measurements when compared to *supervised* measurements (Fig. 2.6), whereas there are cases that *unsupervised* measurements cannot interpret the concepts correctly (Fig. 2.7). From these figures, we have the following findings:

Diversity and domain gap are correlated. Changes in either diversity or domain gap oftentimes affect the other. In data augmentation experiments (Fig. 2.4 top) and neural rendering experiments (Fig. 2.4 middle), created datasets change both diversity and domain gap at the same time. For rotating and translating (Fig. 2.4 bottom), although there are multiple "data points" under the same diversity or domain gap, it is still difficult to directly manipulate the means and variances for the Gaussian distribution so that it *only* influences either domain gap or diversity. All of these indicate the domain gap and diversity are correlated and will be simultaneously influenced by operations.

Neither diversity nor domain gap alone determines the quality of training sets.

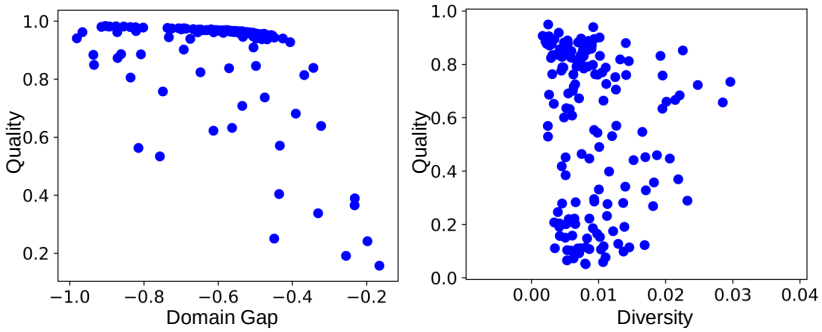

Figure 2.5: Neither diversity nor domain gap alone determines the quality of training sets. **Left:** Relationship between domain gap and training set quality in the MNIST rotation experiment. For a specific value of domain gap, the training set quality can be different in a large range. **Right:** Relationship between diversity and training set quality in the MNIST rotation and translation experiment. Similarly, for a specific value of diversity, the training set quality can be different in a large range.

Many existing works intend to reduce domain gap by a domain adaptation procedure. However, a dataset with the lowest domain gap is not necessarily the best. Similarly, data augmentation is often used to increase the diversity. But a dataset with the highest diversity is also not usually the best. We explain this by making the following statements based on our results. First, from Fig. 2.4, all the top-most points (*i.e.*, training sets with the largest diversity) do not achieve the best results. Similarly, except for MNIST→USPS experiment, all left-most points (*i.e.*, training sets with the lowest domain gap) also do not achieve the best results. Second, from Fig. 2.5, one specific value of domain gap or diversity corresponds to multiple values of training set quality. This indicates that we cannot formulate training data quality as a function of either of these two alone. Instead, a formulation with both concepts included is needed.

A joint objective of high diversity and low domain gap characterizes high-quality training data. In the subfigures of Fig. 2.4 and Fig. 2.6, for fixed values of domain gap, test accuracy generally increases with higher values of diversity. Similarly, for fixed values of diversity, test accuracy generally increases with smaller values of domain gap. As such, we argue that the top left corner (*i.e.*, high diversity and low domain gap) can potentially create the best quality training data.

There exists a trade-off between domain gap and diversity. As discussed above, a low domain gap and high diversity dataset should be the best. This situation is not easy to meet in practice. As mentioned, domain gap and diversity are correlated with each other. For example, in the experiment of geometric image transformation, we find a trade-off between diversity and domain gap: an increase in diversity is often accompanied by an unwanted increase in domain gap. For example, when we increase the transformation scale, diversity and domain gap increase together. In this case, a trade-off between the two values is needed to balance the two objectives and achieve the best result.

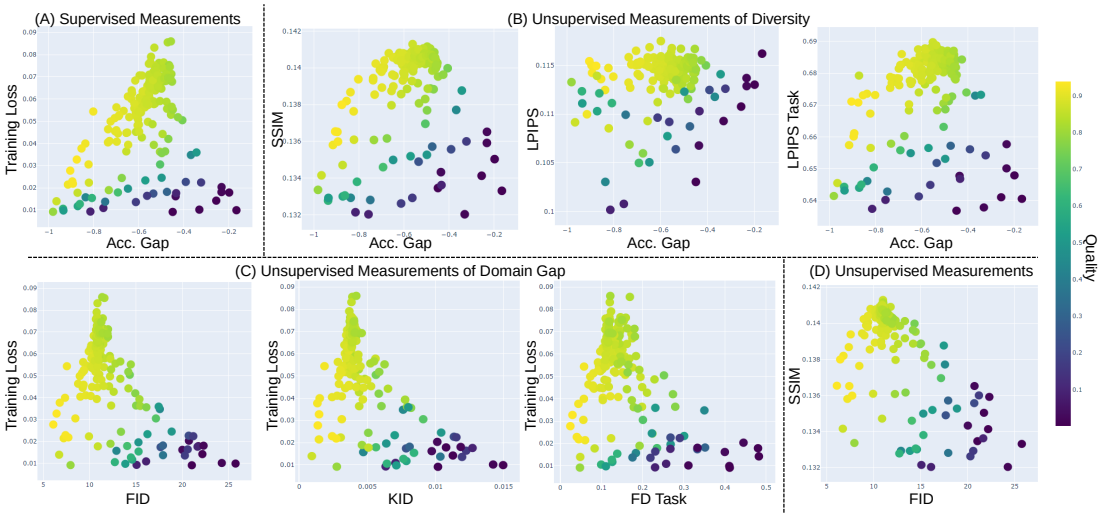

Figure 2.6: Comparison between different *unsupervised* measurements in the MNIST rotation experiment. (A) *supervised* measurements of both domain gap and diversity. (B) *unsupervised* diversity measurements with SSIM, LPIPS and LPIPS Task. (C) *unsupervised* domain gap measurements with FID, KID and FD Task. (D) *unsupervised* measurements of both domain gap and diversity with FID and SSIM.

Unsupervised measurements work well on rotating and translating MNIST. In practice, it is time-consuming to use *supervised* measurements as they require trained models. In this case, we test the *unsupervised* measurements and find they work well in rotation and translation experiments. Fig. 2.6 shows the experiment of MNIST rotation. In this figure, most *unsupervised* measurements have the same trend as *supervised* measurements (*i.e.*, datasets with high diversity and low domain gap achieve good results). One exception here is LPIPS. It seems to consider some datasets with low diversity as high diversity by mistake. We believe the reason is that the model for evaluating LPIPS is trained from human perception [50]. Such human-level understanding is different from model-level understanding. When we switch the model to a task model which is trained on the validation set for the measurements "LPIPS Task", the new "LPIPS Task" seems stable in measuring diversity. Also, *unsupervised* measurements on domain gap seem to consider some datasets with low domain gap as high domain gap. But the trend with domain gap remains the same. We also use "FD Task" which uses a task model to measure distance instead of the Inception-V3 network [51] for FID. We observe a similar trend using FID, KID, and "FD Task".

Unsupervised measurements fail to describe training set quality on neural rendering. Fig. 2.7 shows failure cases. For neural rendering, we observe that the training dataset with a lower domain gap should have better quality (enables higher test performance), which corresponds with our experiments in Fig. 2.4 and Fig. 2.6. However, when using the *unsupervised* measurement for domain gap, *e.g.*, FID, we find there lacks a clear relationship between FID and the quality, indicating the FID measurement might not be suitable. One possible explanation is that the *unsupervised* mea-

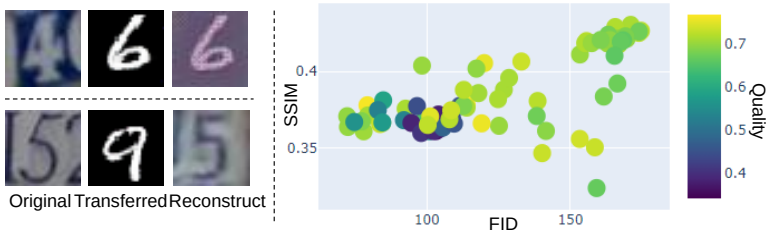

Figure 2.7: Failure cases of *unsupervised* measurements. Content drift situation in neural rendering. **Left**: content drift examples of a style transfer model. For example, the neural rendering is likely to change the image content, making a mismatch between the image and its labels. **Right**: lack of a clear relationship pattern between domain gap and test accuracy.

surement cannot deal with content drift introduced by style transfer (*e.g.*, numbers 4 and 5 are changed into numbers 6 and 9 in Fig. 2.7 (B)).

2.6 Conclusion

Domain gap and diversity are two pillars in describing training set quality. However, they are often separately considered in different situations. This chapter considers training set quality as a function of both domain gap and diversity. We conduct quantitative experiments on sampling datasets with a large range of domain gaps and diversity, for the purpose of searching out their relationships. Mainstream training set creation methods are explored like data augmentation and neural rendering. On these training set creation methods, our findings suggest that a relatively low domain gap and high diversity are desired for good outcomes. As applications of our findings, we proposed the task of the training set search in the next chapter.

Chapter 3

Training Set Search from a Source Pool

3.1 Introduction

The success of a deep learning-based object re-ID solution relies on one of its critical prerequisites: the labeled training data. To achieve high accuracy, typically a massive amount of data needs to be used to train deep learning models. However, creating large-scale object re-ID training data with manual labels is expensive. Furthermore, collecting training data that then contributes to the high test accuracy of the trained model is even more challenging. Recent years have seen a large number of datasets proposed and a significant increase in the data size of any single dataset. For example, the RandPerson [1] dataset has 8,000 identities, which is more than 6$\times$ larger than the previous PersonX [1] dataset.

However, these datasets generally have their own dataset bias, making the model trained on one dataset unable to generalize well to another. For example, depending on the filming scenario, different person re-ID datasets generally have biases on camera positions, race, and clothing style. Such dataset biases usually lead to model bias, which results in the model's difficulty performing well in an unseen filming scenario. To address this, many try to improve learning algorithms, including domain adaptation and domain generalization methods [56, 57, 58, 59, 60, 61]. Whereas these algorithms are well-studied and have proven successful in many re-ID applications, deciding what kind of data to use for training the re-ID model is still an open research problem, and has received relatively little attention in the community. We argue that this is a crucial problem to be answered in light of the ever-increasing scale of the available datasets.

In this chapter, we introduce SnP, a search and pruning solution for sampling an efficient re-ID training set to a target domain. SnP is designed for the scenario in that we have a target dataset that does not have labeled training data. Our collected source pool, instead, provides suitable labeled data to train a competitive model. Specifically, given a user-specified budget (*e.g.*, maximum desired data size), we sample a subset of the source pool, which satisfies the budget requirement and desirably has high-quality data to train deep learning models. This scenario is especially helpful for

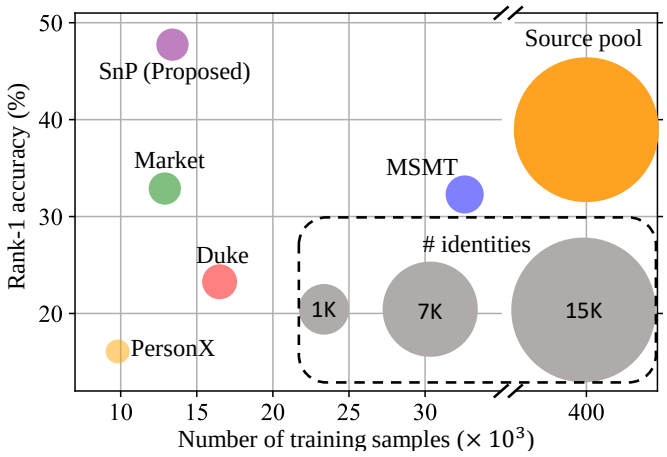

Figure 3.1: We present a search and pruning (SnP) solution to the training data search problem in object re-ID. The source data pool is an order of magnitude larger than existing re-ID training sets in terms of the number of images and the number of identities. When the target is AlicePerson [52], from the source pool, our method (SnP) results in a training set 80% smaller than the source pool while achieving a similar or even higher re-ID accuracy. The searched training set is also superior to existing individual training sets such as Market-1501 [53], Duke [54], and MSMT [55].

deploying a re-ID system for unknown test environments, as it is difficult to manually label a training set for these new environments. We note that due to the absence of an in-distribution training set, the searched data are directly used for training the re-ID model rather than pre-training.

In particular, we combine several popular re-ID datasets into a source pool, and represent each image in the pool with features. Those features are extracted from an Imagenet-pretrained model [51]. The images with features are stored on the dataserver to serve as a gallery. When there is a query from the target, we extract the feature of the query image, and search in the gallery for similar images on a feature level. Specifically, in the search stage, we calculate feature-level distance, *i.e.*, Fréchet Inception Distance (FID) [28]. Given the constraint of a budget, we select the most representative samples in the pruning stage, based on the outputs from the search stage. This limits the size of the constructed training set and enables efficient training.

Combining search and pruning, we construct a training dataset that empirically shows significant accuracy improvements on several object re-ID targets, compared to the baselines. Without budget constraints, our searched training sets allow higher re-ID accuracy than the complete source pool, due to its target-specificity. With budget constraints, the pruned training set still achieves comparable or better performance than the source pool. The proposed SnP is demonstrated to be superior to random or greedy sampling. We show in Fig. 3.1 that the training set constructed by SnP leads to the best performance on the target compared to the others. We provide discussions on the specificity of our method and its role in bridging the re-ID domain gap.

3.2 Related Work

Active learning gradually searches over unlabeled data to find samples to be labeled by the oracle [62]. It is an iterative training process, where data search is performed at each iteration to train a task model. Our task is different from active learning. First, active learning is designed to acquire a training set and train models gradually. This requires multiple real training processes which are computationally expensive. In comparison, we directly search the whole training set once. Then the searched training set is used for model training. Second, many active learning methods require access to the target task model to provide selection metrics, *e.g.*, uncertainty-based metrics [63, 64, 65, 66]. Our task does not require task model information, in other words, it is a real training process, during the process of the training set search. Thus enabling a fast process of getting training data.

Neural data server is the closest inspiring work [67, 62]. They also aim to search training data all at once from a large database. However, compared with us, firstly, [67] and [62] are designed for searching pretraining data rather than direct training data. Such a design is understandable as they are mainly for the classification task. Searching directly for training data requires careful class alignment. In comparison, We are targeting the re-ID task, where its training set can have a different class from the target, and then be directly used for training models. Furthermore, [67] and [62] require unsupervised pretrained experts to measure the domain gap. Meanwhile we do not require this, which saves extraction time and simplifies the solution.

Transfer learning is a long-standing problem for re-ID tasks, and many attempts at learning algorithms have been made to reduce the effect of domain gap [38, 68, 69, 36, 70]. Common strategies contain feature-level [34] and pseudo-label based [56, 57, 58] domain adaptation, and domain generalization [59, 60, 61]. In this chapter, we focus on training data that is orthogonal to existing training algorithms. As will be shown in experiments, together with some domain adaptation (*i.e.*, pseudo-label) methods, SnP can achieve higher re-ID accuracy.

Learning to generate synthetic training data. Data simulation is an inexpensive alternative to increasing training set scale while providing accurate image labels [2, 71, 1, 72, 73]. These methods aim to lower the domain gap between synthetic data and real data by searching a set of parameters that control the 3D rendering process [2, 71]. In comparison, our search is not conducted on predefined parameters but on data directly. This enables more direct research on how to form a good training set.

Object re-ID has received increasing attention in the past few years, and many effective systems have been proposed [74, 75, 76, 77]. In this thesis, we study object re-ID datasets rather than algorithms. Depending on the camera conditions, location and environment, existing object re-ID datasets usually have their own distinct characteristics or bias [1, 71, 78]. We show details in §3.3.

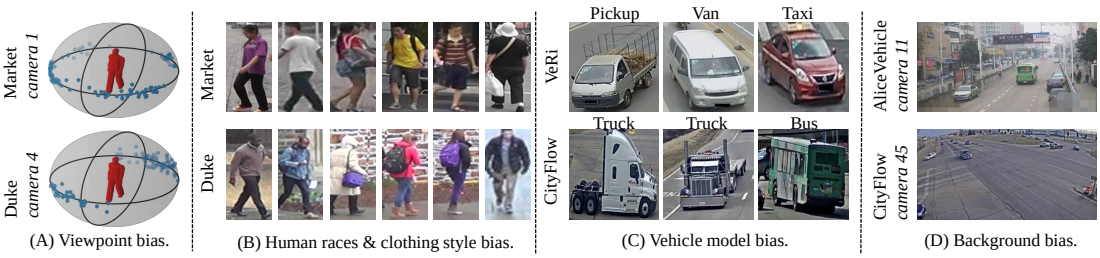

Figure 3.2: Dataset bias in existing object re-ID datasets. (A) Viewpoint distribution of different re-ID datasets. Each blue dot indicates a sample filmed from a specific viewpoint. Compared with the bi-modal viewpoint distribution in Market camera 1, Duke camera 4 has more diverse viewpoints. (B) Different races and clothing styles of different person re-ID datasets. (C) Different vehicle makes in different datasets. For example, taxis in VeRi are unlikely to be found in CityFlow. (D) Different backgrounds in different datasets. We show a city background in AliceVehicle versus an urban background in CityFlow. To tackle such bias in the target domain, we design an automatic way to generate a training set with similar distribution or bias.

3.3 Motivation: Tackling Target Bias

Data bias commonly exists in the re-ID datasets (for examples see below) [72, 79, 80, 81], and it becomes problematic when the training and testing have different biases. Given a target domain with a certain bias, we aim to find a target-specific training set that has similar distribution or bias. Depending on the filming scenario, there are four major types of data biases in existing re-ID datasets. We show examples of each type in Fig. 3.2.

Viewpoint. Viewpoint bias applies to generic objects, including persons and vehicles. We visualize the viewpoint distributions of two representative person re-ID datasets in Fig. 3.2(A), *i.e.*, Market-1501 [53] (denoted as Market) and Duke-reID [54] (denoted as Duke)[1].

Race and clothing style. Subject to the places where the data is collected, identities in the person re-ID datasets can have distinctive patterns. We show in Fig. 3.2(B) that the humans in Market and Duke datasets evince distinct races and clothing styles.

Vehicle model. Similar to the identity bias (*i.e.*, race and clothing style) in person re-ID, the vehicle identities in vehicle re-ID also hold distinct patterns across different datasets. We show examples in Fig. 3.2(C) using the VeRi [82] (denoted as VeRi) and the CityFlow [83].

Background. Background bias exists in both person and vehicle re-ID datasets, which is similar to viewpoint bias. Fig. 3.2(D) compares the background difference of images from two different vehicle re-ID datasets.

[1]We understand that it is no longer encouraged to use the Duke dataset. In fact, we are not using it for algorithm design, but moving forward to find solutions to replace such specific dataset use.

3.4 Method

Given a target unlabeled dataset, we aim to construct a source labeled dataset that has minimal data bias inconsistencies with the target, under certain budget constraints. It induces the model trained on the source to show good performance on the target. To achieve the construction of the source (training) dataset, we propose the search and pruning (SnP) framework.

3.4.1 Overview

We denote the target set as $\mathcal{D}_T = \{(\boldsymbol{x}_i, y_i)\}_{i\in[m_t]}$ where m_t indicates the number of image-label pairs in the target and $[m_t] = \{1, 2, \ldots, m_t\}$. It follows the distribution p_T, *i.e.*, $\mathcal{D}_T \sim p_T$. Let $\mathcal{D}_S$ be the source set to be constructed under a budget b. The budget is specified by the number of identities n and the number of images m allowed in $\mathcal{D}_S$, denoted as $b = (n, m)$. A high budget can lead to an unwanted increase in the training cost, in terms of either training time or model size.

To construct the training set $\mathcal{D}_S$, we create a source pool $\mathcal{S}$, which is a collection of multiple object re-ID datasets. It is represented as $\mathcal{S} = \mathcal{D}_S^1 \bigcup \mathcal{D}_S^2 \cdots \bigcup \mathcal{D}_S^K$. Here each $\mathcal{D}_S^k$, $k \in [K]$, indicates the k-th source re-ID dataset. Given the source pool, we firstly build a subset $\mathbf{S}^*$ of $\mathcal{S}$ regardless of the budget constraint. Let $h_{\mathbf{S}}$ be a model h trained on an arbitrary dataset $\mathbf{S}$. The prediction risk of $h_{\mathbf{S}}$ on the test sample $\boldsymbol{x}$ with ground truth label y is computed as $\ell(h_{\mathbf{S}}(\boldsymbol{x}), y)$. We build $\mathbf{S}^*$ by ensuring that the model $h_{\mathbf{S}^*}$ has minimized risk on $\mathcal{D}_T$, *i.e.*,

$$\mathbf{S}^* = \underset{\mathbf{S}\in 2^{\mathcal{S}}}{\arg\min}\, \mathbb{E}_{\boldsymbol{x},y\sim p_T}[\ell(h_{\mathbf{S}}(\boldsymbol{x}), y)]. \tag{3.1}$$

We apply target-specific search in §3.4.3 to construct $\mathbf{S}^*$.

It can be seen that the construction of $\mathbf{S}^*$ does not take the budget constraint $b = (n, m)$ into consideration, which is otherwise important in reality. Therefore, we build the training set $\mathcal{D}_S$ by pruning $\mathbf{S}^*$ to comprise no more than n identities and no more than m images. Details of the budgeted pruning process are introduced in §3.4.4.

3.4.2 Correlation Study

In order to know how to obtain $\mathbf{S}^*$, we conduct the correlation study to learn the relationships between the dataset bias differences (*i.e.*, domain gap measured in FID [28]), the number of IDs and the training set quality (*i.e.*, rank-1 test accuracy with the model trained on such training set). Shown in Fig. 3.3, in each subfigure, each point represents a training set, which is clustered from the source pool. For each training set, we calculate its domain gap to the target domain, count its number of IDs, and evaluate the rank-1 accuracy. We use the Pearson correlation coefficient (γ) [84] to measure the correlation between them. The Pearson correlation coefficients range from $[-1, 1]$. A value closer to -1 or 1 indicates a stronger negative or positive correlation, respectively, and 0 implies no correlation.

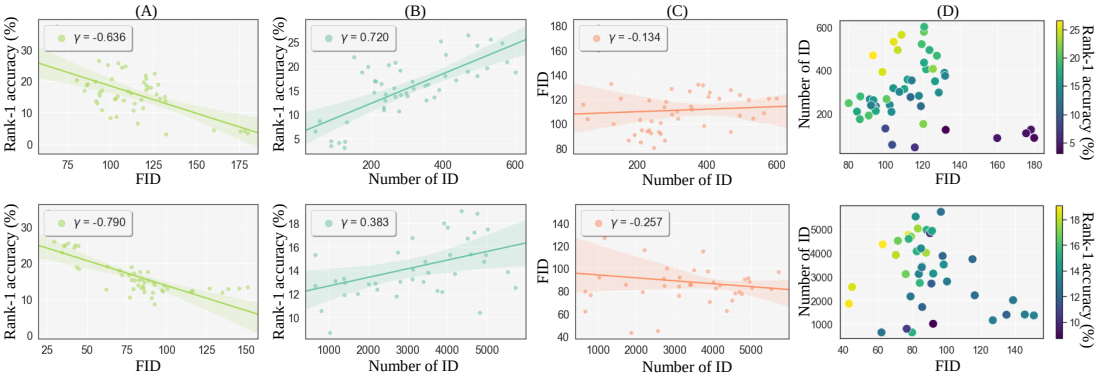

Figure 3.3: Relationships between the domain gap (measured using FID), the number of ID and the rank-1 accuracy on the target. **Top**: correlations when AlicePerson [52] is targeted. **Bottom**: correlations when AliceVehicle [52] is targeted. The Pearson correlation is used to measure the relationship between them. (A) FID *vs*. the rank-1 accuracy. They have a relatively strong negative correlation (≤ -0.636). (B) The number of ID *vs*. the rank-1 accuracy. There exists a positive correlation but such a correlation is not stable. (C) FID *vs*. number of ID. The correlation is weak between them. (D) The joint influence of FID and the number of ID on rank-1 accuracy. The top left corner, *i.e.*, training sets that have low FID scores and a large number of IDs have high rank-1 accuracy on the target.

From this Fig. 3.3(A), we observe a relatively strong negative correlation between domain gap and rank-1 accuracy. This indicates minimizing the domain gap between the source and target is highly likely to improve training set quality, *i.e.*, test set rank-1 accuracy. From this Fig. 3.3(B), we observe a positive but unstable correlation between the number of ID and the rank-1 accuracy. For example, when AlicePerson is targeted, they have a relatively strong positive correlation (0.720). However, such correlation is only 0.383 when AliceVehicle is targeted. Fig. 3.3(C) shows the correlation between FID and the number of ID is weak. They are independent factors that influence training set quality. Fig. 3.3(D) shows the joint influence of FID and the number of ID on the training set quality. The top left corner in Fig. 3.3(D), *i.e.*, training sets that have a low domain gap to the target and a large number of IDs are of high quality, thereby can train a model that has higher rank-1 accuracy on the target.

3.4.3 Target-specific Subset Search

The theory of domain adaptation [85] states that:

$$\varepsilon_T(h) < \varepsilon_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}, \mathcal{D}_T). \tag{3.2}$$

Here $h \in \mathcal{H}$ represents the hypothesis function (*i.e.*, the model). $\varepsilon_T(h)$ is the risk of model h on the target set $\mathcal{D}_T$, while $\varepsilon_S(h)$ is its risk on the source set $\mathbf{S}$. $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}, \mathcal{D}_T)$ is the unlabelled $\mathcal{H}\Delta\mathcal{H}$ divergence [85] between $\mathbf{S}$ and $\mathcal{D}_T$. Equation 3.2 shows that the target risk $\varepsilon_T(h)$ is upper bounded by $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}, \mathcal{D}_T)$.

In the common practice of feature-level domain adaptation, the source training set $\mathbf{S}$ is fixed while the joint feature extraction model is used to minimize $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}, \mathcal{D}_T)$ [34]. In our design, the feature extraction model is fixed instead. To minimize $\varepsilon_T(h)$, the problem is reformulated as

$$\mathbf{S}^* = \arg\min_{\mathbf{S}\in 2^{\mathcal{S}}} d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}, \mathcal{D}_T). \tag{3.3}$$

Generally, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbf{S}^*, \mathcal{D}_T)$ is difficult to compute, but many alternatives exist in the literature [34]. We use Fréchet Inception Distance (FID) [28], which is defined as:

$$\mathrm{FID}(\mathbf{S}^*, \mathcal{D}_T) = \|\boldsymbol{\mu}_s - \boldsymbol{\mu}_t\|_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_s + \boldsymbol{\Sigma}_t - 2(\boldsymbol{\Sigma}_s\boldsymbol{\Sigma}_t)^{\frac{1}{2}}). \tag{3.4}$$

In Eq. 4.4, $\boldsymbol{\mu}_s \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_s \in \mathbb{R}^{d\times d}$ are the mean and covariance matrix of the image descriptors of $\mathbf{S}^*$, respectively. $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ are those of $\mathcal{D}_T$. $\mathrm{Tr}(.)$ represents the trace of a square matrix. d is the dimension of the image descriptors. Consequently, the objective function is reduced as,

$$\mathbf{S}^* = \arg\min_{\mathbf{S}\in 2^{\mathcal{S}}} \mathrm{FID}(\mathbf{S}, \mathcal{D}_T). \tag{3.5}$$

We build $\mathbf{S}^*$ with the greedy algorithm below.

Firstly, we divide the entire dataset $\mathcal{S}$ into J clusters $\{\mathbf{S}^1, \cdots, \mathbf{S}^J\}$, as shown in Fig. 3.4 (A). Specifically, we average all image descriptors that belong to the same identity, and use this ID-averaged descriptor to represent all corresponding images. Afterwards, we cluster the ID-averaged features into J groups using the k-means method [86]. Each subset $\mathbf{S}^j$, $j \in [J]$, is composed of all images with the corresponding IDs in that group. Secondly, we calculate the FID between each subset $\mathbf{S}^j$ and the target $\mathcal{D}_T$, and sort $\{\mathrm{FID}(\mathbf{S}^j, \mathcal{D}_T)\}_{j\in[J]}$ in ascending order. To build $\mathbf{S}^*$, we keep adding the subsets with lower FID to $\mathbf{S}^*$ until $\mathrm{FID}(\mathbf{S}^*, \mathcal{D}_T)$ stops to decrease. This indicates that the constructed $\mathbf{S}^*$ holds the minimum FID to the target set. Algorithm 1 summarizes the above procedures. We empirically demonstrate in Fig. 3.5 that, the subset $\mathbf{S}^*$ with the minimum $\mathrm{FID}(\mathbf{S}^*, \mathcal{D}_T)$ results in the model to produce highest accuracy on the target dataset.

3.4.4 Budget-constrained Pruning

Using the target-specific subset search, we construct a candidate training set $\mathbf{S}^*$. For now, it can violate the budget constraint of $b = (n, m)$, which prefers the training set to include no more than n identities and no more than m images. Let $\mathcal{Y}(\mathbf{S}^*) = \{y^1, y^2, \ldots, y^a\}$, be the set of unique identities in $\mathbf{S}^*$, and $\boldsymbol{s}(y^i)$, $i \in [a]$ be the set of all images with identity y^i. Generally, we have $a \geq n$. To get the training set $\mathcal{D}_S$ under budget b, we randomly sample a subset $\boldsymbol{y}$ of n identities from $\mathcal{Y}$. All images with identities in $\boldsymbol{y}$ are combined to form a subset $\hat{\mathbf{s}} = \{\boldsymbol{s}(y^i)|y^i \in \boldsymbol{y}\}$. We also initialize $\mathcal{D}_S$ by random sampling a seed image from each $\boldsymbol{s}(y^i)$, $y^i \in \boldsymbol{y}$. This guarantees $\mathcal{D}_S$ to cover all the identities in the subset $\hat{\mathbf{s}}$.

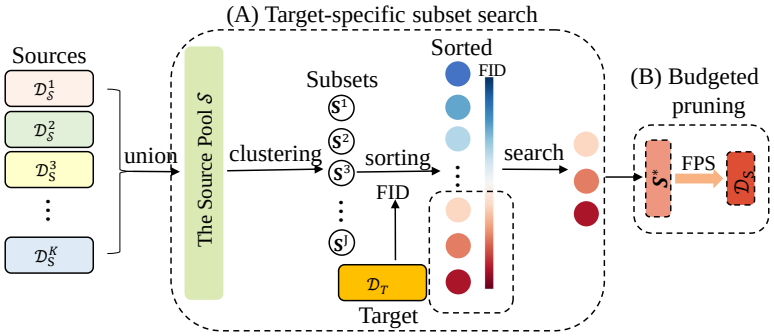

Figure 3.4: Workflow of the proposed SnP method. We are given sources composed of K existing datasets. From these sources, we aim to construct a training set, which satisfies the budget of no more than n IDs and m images. To achieve this, we perform (A) target-specific subset search to obtain a subset $\mathbf{S}^*$ with similar distributions to the target, then perform (B) budgeted pruning to select n IDs and m representative images, forming the final training set.

Suppose the number of images in $\hat{\mathbf{s}}$ is larger than m. We construct $\mathcal{D}_S$ by sampling images iteratively from $\hat{\mathbf{s}}$ until $|\mathcal{D}_S| = m$. To ensure similar performance between the model trained on D_S and the model trained on $\hat{\mathbf{s}}$, we minimize the risk differences between them,

$$\mathcal{D}_S = \arg\min_{\mathcal{D}_S \in 2^{\hat{\mathbf{s}}}} \left| L(h_{\hat{\mathbf{s}}}(\boldsymbol{x}), y) - L(h_{\mathcal{D}_S}(\boldsymbol{x}), y) \right|. \tag{3.6}$$

$L(h_{\hat{\mathbf{s}}}(\boldsymbol{x}), y)$ and $L(h_{\mathcal{D}_S}(\boldsymbol{x}), y)$ are the respective risks of model h on the dataset $\hat{\mathbf{s}}$ and $\mathcal{D}_S$. For explicity, we define the risk of model h on an arbitrary dataset $\mathbf{S}$ as

$$L(h_{\mathbf{S}}(\boldsymbol{x}), y) = \frac{1}{|\mathbf{S}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathbf{S}} \ell(h_{\mathbf{S}}(\boldsymbol{x}_i), y_i). \tag{3.7}$$

As in Eq. 3.1, $\ell(h_{\mathbf{S}}(\boldsymbol{x}_i), y_i)$ is the risk on individual samples.

From the theory of core set [87], if $\mathcal{D}_S$ is the δ_s cover of the set $\hat{\mathbf{s}}$ and shares the same number of classes with $\hat{\mathbf{s}}$, the risk difference between model $h_{\hat{\mathbf{s}}}$ and $h_{\mathcal{D}_S}$ is bounded by

$$\left| L(h_{\hat{\mathbf{s}}}(\boldsymbol{x}), y) - L(h_{\mathcal{D}_S}(\boldsymbol{x}), y) \right| \leq \mathcal{O}(\delta_s) + \mathcal{O}(\frac{1}{\sqrt{|\mathcal{D}_S|}}). \tag{3.8}$$

δ_s is the radius of the cover, and $\mathcal{O}(\delta_s)$ is a polynomial function over δ_s. The problem can be reduced as a K-center problem [88] by optimizing $\mathcal{O}(\delta_s)$. We apply a 2-approximation algorithm [89] to iteratively find optimal samples in $\hat{\mathbf{s}}$ and add to $\mathcal{D}_S$. Specifically, each optimal sample $\mathbf{z}^*$ is computed as

$$\mathbf{z}^* = \arg\max_{\mathbf{z}_i \in \hat{\mathbf{s}} \setminus \mathcal{D}_S} \min_{\mathbf{z}_j \in \mathcal{D}_S} \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|_2, \tag{3.9}$$

where $\mathbf{z} = (\boldsymbol{x}, y)$, and $f(\boldsymbol{x})$ represents the descriptor of an image $\boldsymbol{x}$. Equation 3.9 relates to the furthest point sampling method [90], which enables the most representative samples from a dataset to be selected. We summarize the prunning process in

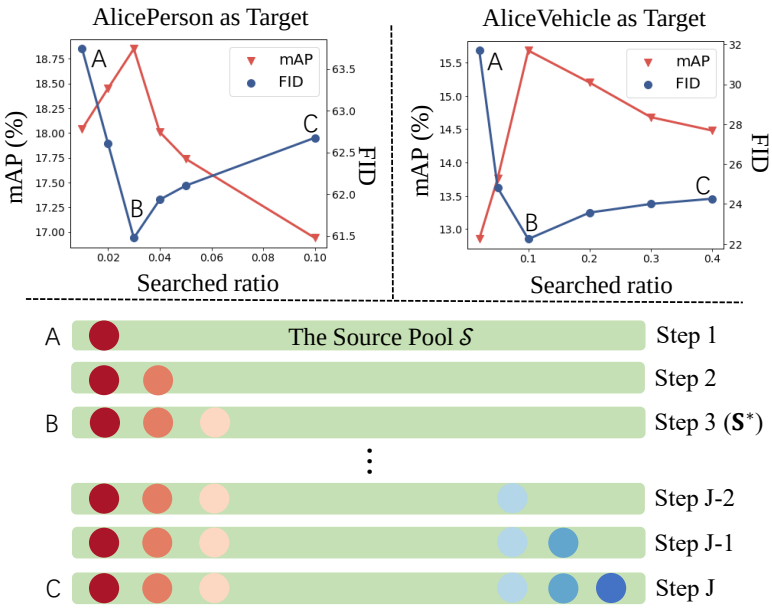

Figure 3.5: Performance of constructed training set at different iterations on the target set. We greedily add clustered subsets to form $\mathbf{S}^*$ which has the smallest domain gap to the target $\mathcal{D}_T$. During this process, we have relatively high FID and low accuracy at (A). Gradually, we reach situation (B) of the smallest domain gap and highest accuracy. After (B), adding subsets leads to an increase in the domain gap, and the accuracy drops, *e.g.*, (C). To study the correlation between FID and mAP during the search process, we hope to eliminate the impact of dataset size. Thus we use the same number of IDs from the searched set (*i.e.*, 2% ID).

Algorithm 2.

3.4.5 Discussion

What to prune first, ID or image? In our design, we select the IDs first and then images, aiming to build a dataset that has a small domain gap to target and is meanwhile small in scale. Selecting the images first and then IDs is possible. However, it leads to a significant increase in the time complexity. As shown in Algorithm 2, the most time-consuming part of our algorithm is the FPS for finding image core sets, which takes $\mathcal{O}(m|\hat{\mathbf{s}}|)$. If we select images first, the time complexity becomes $\mathcal{O}(m|\mathbf{S}^*|)$, which is significantly higher than $\mathcal{O}(m|\hat{\mathbf{s}}|)$ as $|\mathbf{S}^*| \gg |\hat{\mathbf{s}}|$.

Is SnP applicable to tasks beyond object re-ID? It is possible to apply our search and pruning (SnP) framework to other tasks, *e.g.*, classification. Yet, achieving this requires the SnP algorithm to be partially redesigned. This is because in object re-ID, the major domain gap comes from the class difference, *i.e.*, IDs. However, such a gap does not exist in classification tasks as the train/val/test datasets always share the same classes. Therefore, our current design of selecting similar IDs for similar distributions is not directly applicable, and has to be adapted.

Algorithm 1 Target-specific Subset Search

Input: Source pool $\mathcal{S}$, target set $\mathcal{D}_T$, and the number of clusters J.
Begin:
Cluster $(\mathcal{S}, J) \longrightarrow \{\mathbf{S}^1, \cdots, \mathbf{S}^J\}$
$\mathbf{s} = \emptyset, \epsilon = \infty, \mathbf{S}^* = \emptyset$
$\mathcal{J} = \text{argsort}\big(\{\text{FID}(\mathbf{S}^j, \mathcal{D}_T)\}_{j\in[J]}\big)$ ▷ Ascending
for j in $\mathcal{J}$ **do**
    $\mathbf{s} = \mathbf{s} \cup \mathbf{S}^j$
    if $\text{FID}(\mathbf{s}, T) < \epsilon$ **then**
        $\epsilon = \text{FID}(\mathbf{s}, \mathcal{D}_T)$
        $\mathbf{S}^* = \mathbf{s}$
    end if
end for
return $\mathbf{S}^*$

3.5 Experiment

We evaluate the effectiveness of SnP on person re-ID and vehicle re-ID. In both tasks, given target data, we use the SnP pipeline to find a training set that has a similar distribution to the target and simultaneously meets a budget.

3.5.1 Source and Target Datasets

Person re-ID. We create the source pool for person re-ID using 10 public datasets, including Market [53], Duke [54], MSMT17 [55] (denoted as MSMT), CUHK03 [91], RAiD [92], PersonX [1], UnrealPerson [72], RandPerson [79], PKU-Reid [93] and VIPeR [94]. Those datasets cover both synthetic and real-world data. Ten re-ID datasets constitute a source pool that contains in total of 15,060 IDs and 399,715 images.

We use two real-world datasets as targets: AlicePerson [52] and Market [53]. Specifically, AlicePerson is specially designed for domain adaptation as it contains unlabeled training images. Thus it is only used for targets and not included in the source pool Note that in Market, the label for its training set is not be used. When Market is the target, Market training set is excluded from the source pool.

Vehicle re-ID. For vehicle re-ID, we create the source pool by combining 8 datasets, which are VeRi [82], CityFlow [83], VehicleID [3], VeRi-Wild [81], VehicleX [2], Stanford Cars [95], PKU-vd1 [80] and PKU-vd2 [80]. It totally has 156,512 IDs and 1,284,272 images.

AliceVehicle [52] and VeRi [82] are separately used as the target domains. Similar to AlicePerson, AliceVehicle is also designed for domain adaptation, and used as the target only. When VeRi is the target, the VeRi training set is excluded from the source pool.

Evaluation protocol. 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,

Algorithm 2 Budgeted Pruning

Input: Initial source set $\mathbf{S}^*$, budget $b = (n, m)$.
Begin:
$\boldsymbol{y} = \mathcal{U}(\mathcal{Y}(\mathbf{S}^*), n)$ ▷ Sample n IDs
$\hat{\mathbf{s}} = \{\boldsymbol{s}(y_i) | y_i \in \boldsymbol{y}\}$
$\mathcal{D}_S = \{\mathcal{U}(\mathbf{s}(y^i), 1) | y_i \in \boldsymbol{y}\}$
if $|\hat{\mathbf{s}}| > m$ **then** ▷ Sample m images
  repeat
    $\mathbf{z}^* = \arg\max_{\mathbf{z}_i \in \hat{\mathbf{s}} \setminus \mathcal{D}_S} \min_{\mathbf{z}_j \in \mathcal{D}_S} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$
    $\mathcal{D}_S = \mathcal{D}_S \cup \{\mathbf{z}^*\}$
  until $|\mathcal{D}_S| = m$
else
  $\mathcal{D}_S = \hat{\mathbf{s}}$
end if
return $\mathcal{D}_S$

and "rank-5" means the success rate of ranking at least one true match within top-5 matches.

3.5.2 Experimental Details

SnP settings. For a dataset denoted as a target domain (*e.g.*, Market and VeRi), their unlabeled training sets are used as the search target. For image feature extraction, we use IncepetionV3 [51] pretrained on ImageNet [10].

Task model configuration. We have two types of models: direct transfer models and pseudo-label based models. For direct transfer, we use multiple task models, including ID-discriminative embedding (IDE) [96], the part-based convolution (PCB) [97], and TransReid [98]. For IDE, we adopt the strategy from [99] which uses ResNet-50 [13], adds batch normalization and removes ReLU after the final feature layer. For PCB, we use the ResNet-50 backbone and vertically partition an image into six equal horizontal parts. For pseudo-label methods, we use UDA [100] and MMT [101]. The unlabeled target training set (*e.g.*, AlicePerson training set) is used for pseudo-label model training. Note that when implementing these task models, we use their official implementation with default hyperparameters including learning rate and training epochs.

Computation Resources. We run experiments on an 8-GPU machine and an oracle cloud machine. The 8-GPU machine has 8 NVIDIA 3090 GPUs and its CPU is AMD EPYC 7343 Processor. For the oracle cloud machine, it has 2 NVIDIA A10 GPUs and its CPU is Intel Platinum 8358 Processor. For both the 3090 machine and the oracle cloud machine, we use PyTorch version 1.12.1+cu116.

Table 3.1: Comparing different methods in training data search: SnP, random sampling, and greedy sampling. We set the budget as 2%, 5%, and 20% of the total source IDs. We use four targets: AlicePerson, Market, AliceVehicle and VeRi. The task model is IDE [96]. FID, rank-1 accuracy (%), and mAP (%) are reported.

Training data			Person re-ID targets						Vehicle re-ID targets					
			AlicePerson			Market			AliceVehicle			VeRi		
			FID↓	R1↑	mAP↑	FID↓	R1↑	mAP↑	FID↓	R1↑	mAP↑	FID↓	R1↑	mAP↑
Source pool			81.67	38.96	17.62	37.53	55.55	30.62	43.95	30.47	14.64	24.39	55.90	25.03
Searched			60.95	48.19	25.51	30.42	61.49	34.40	23.44	46.78	25.46	17.92	74.13	41.71
Pruned	2% IDs	Random	80.06	23.67	9.80	39.03	40.77	19.54	45.97	31.35	12.87	24.68	67.16	26.48
		Greedy [67]	61.78	33.22	15.10	34.43	42.19	19.35	38.51	31.82	12.91	29.53	66.57	26.24
		SnP	60.42	38.23	18.17	31.29	44.80	21.65	23.48	38.48	17.79	17.70	69.96	31.10
	5% IDs	Random	81.41	33.16	14.49	39.65	47.39	23.97	44.52	36.36	14.17	25.27	70.38	30.44
		Greedy [67]	61.01	44.63	22.81	31.63	49.17	24.77	32.48	41.32	17.77	26.06	71.23	32.53
		SnP	60.64	47.26	25.45	30.37	51.96	26.56	23.92	44.58	21.79	18.09	72.05	36.01
	20% IDs	Random	79.33	38.10	17.79	38.63	53.15	28.39	43.90	40.89	18.13	24.43	68.71	34.10
		Greedy [67]	63.15	46.74	22.65	32.42	53.53	28.19	24.15	44.58	22.82	18.74	71.04	38.07
		SnP	61.87	47.20	25.36	30.58	57.14	33.09	23.47	46.07	25.24	17.93	73.48	40.75

Table 3.2: The effectiveness of the target-specific subset search. mAP (%) and CMC scores are reported. "R1" and "R5" denote rank-1 accuracy (%) and rank-5 accuracy (%), respectively.

	AlicePerson				AliceVehicle		
Training data	R1	R5	mAP	Training data	R1	R5	mAP
Market [53]	32.89	52.54	16.06	VeRi [82]	30.69	43.66	11.05
Duke [54]	23.27	41.13	8.59	CityFlow [83]	23.95	36.00	7.45
MSMT [55]	31.29	52.79	15.08	VehicleID [3]	16.3	29.13	4.73
PersonX [1]	16.08	27.42	6.41	VehicleX [2]	18.85	32.18	8.89
Source pool	38.96	57.48	17.62	Source pool	30.47	48.33	14.64
Searched	48.19	67.57	25.51	Searched	46.78	64.41	25.46

3.5.3 Results

SnP framework *vs.* random sampling and greedy sampling. Given a target domain, the SnP framework allows us to construct a budgeted dataset with a similar distribution. We show the superiority of SnP framework in Table 3.1. After sampling target-specific data with SnP, we train the subsequent re-ID model with the sampled target-specific data only. In Table 3.1, we compare the sampled dataset with those created by greedy sampling and random sampling. Random sampling means we randomly select IDs according to a uniform distribution. For greedy sampling, we reproduce [67]. Specifically, We assign each ID a score, which is calculated using FID. IDs will then be selected greedily from the lowest FID value to the highest FID value.

From the results shown in Table 3.1 and Table 3.3, we observe that training sets selected using SnP achieve consistently lower domain gap to the target and higher re-ID accuracy than those found by random sampling and greedy sampling. For example, when creating a training set with only 2% source IDs for AlicePerson as the target, SnP results in -19.64 and -1.36 improvement in FID value, and +14.56% and +5.01% improvement in rank-1 accuracy over using random sampling and greedy sampling, respectively. When sampling training set with 2% (301 out of 15,060 IDs in the source

Table 3.3: The superiority of SnP over random sampling and greedy sampling, when different direct transfer and pseudu-label re-ID models are used. We report accuracy when 2% IDs are selected for target AlicerPerson. Notations and evaluation metrics are the same as those in the previous table.

Type	Model	Training data	R1	R5	mAP
Direct Transfer	IDE [96]	Random	23.67	42.32	9.80
		Greedy [67]	33.22	54.45	15.10
		SnP	38.23	58.40	18.17
	PCB [97]	Random	24.79	41.60	9.91
		Greedy [67]	29.07	46.01	12.59
		SnP	32.43	49.90	15.18
	TransReid [98]	Random	52.04	69.73	28.47
		Greedy [67]	63.73	80.24	41.88
		SnP	64.31	80.46	42.74
Pseudo -labeling	UDA [100]	Random	32.17	54.67	15.32
		Greedy [67]	36.47	52.06	17.34
		SnP	41.41	55.74	20.47
	MMT [101]	Random	35.94	51.91	17.25
		Greedy [67]	38.64	56.48	21.18
		SnP	43.36	60.38	23.34

pool) of the source IDs for AliceVehicle as the target, the rank-1 accuracy improvement is +7.13% and +6.66%, respectively.

Effectiveness of the search step. We analyze this step in Table 3.1 and Table 3.2. In this search process that poses no limit on the number of IDs and images, we aim to identify and merge clusters of source identities that exhibit similar distributions with the target. We show the searched datasets contribute to improved accuracy over the entire data pool. For example, for AlicePerson, we observe a +9.23% improvement in rank-1 accuracy over the entire data pool. We further show the searched data is superior to individual training sets in Table 3.2. For example, when AliceVehicle is our target, we show our searched dataset results in +16.09%, +22.83%, +30.48%, +27.93% higher rank-1 accuracy than VeRi, CityFlow, VehicleID, and VehicleX, respectively.

Of note, accuracy under this application scenario is usually lower than that produced by in-distribution training sets. This difference is understandable, because searched data have a relatively lower resemblance to the target data compared with in-distribution training sets. That said, annotating in-distribution training sets is usually expensive, especially considering the complex, specific and ever-changing target environments, where creating a training set on-the-fly with good performance is of practical value.

Effectiveness of the pruning step is analyzed in Table 3.1 and Fig. 3.7. Pruning aims to find a subset that has no more than n IDs and m images. From Table 3.1 and Fig. 3.7, admittedly, the pruning of both IDs and images will lead to an accuracy decrease. For example, when the target is AlicePerson, if we select 2% IDs, there is a -9.23% decrease in rank-1 accuracy. From Fig. 3.7, when the target is AlicePerson, if we further select 40% images, there is a -9.96% decrease in rank-1 accuracy. However, even though only 2% IDs are used, rank-1 accuracy obtained from the pruned training data is still competitive over the source pool, which is just -0.73% lower than the

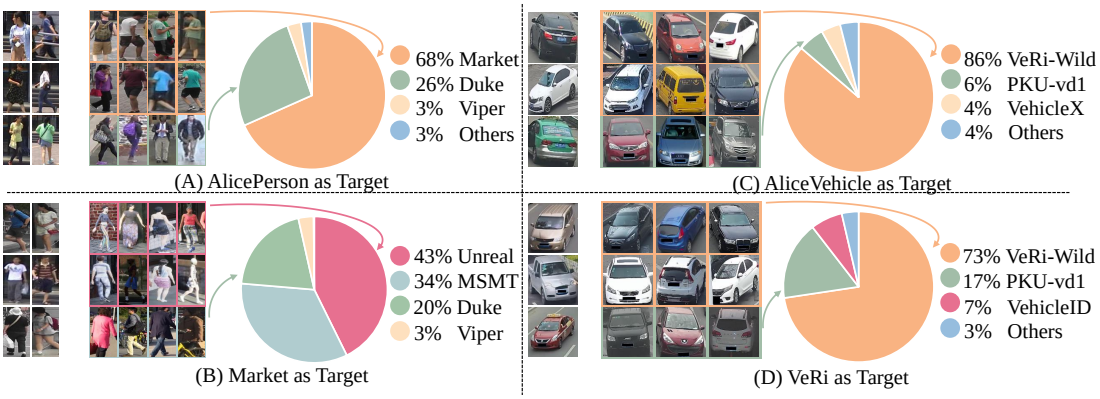

Figure 3.6: Composition statistics and images samples of searched training sets. For each subfigure (A), (B), (C) and (D), the columns on the left present unlabeled target samples; the middle columns provide samples of the searched training set; the pie chart on the right shows composition statistics of the searched training set.

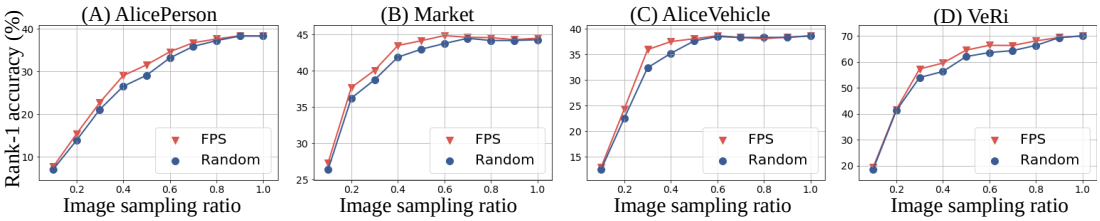

Figure 3.7: Comparing FPS with random sampling. The search step provides us with 2% of the source IDs. We further select various ratios of images from these IDs. Four different targets are used, and rank-1 accuracy is the evaluation metric. Score is averaged over three runs.

source pool. It shows the pruning method significantly reduces the training set scale while being able to train a model of reasonable accuracy.

Composition of searched training sets. We visualize four examples in in Fig. 3.6. It is clear that searched training sets have different compositions under different targets. If we use AlicePerson as the target, in the searched training set, images from Market and Duke take up 68% and 26%, respectively. In comparison, when using Market as the target, the resulting training set contains, 43% images from UnrealPerson, and 34% from MSMT. An interesting observation is that synthetic data (UnrealPerson) has a major role under Market as the target. It demonstrates the value of the use of synthetic data for real-world target domains.

Comparison between FPS and random sampling. Both can be used for sampling images resulting from the search step. In Fig. 3.7, we sample different ratios of the images resulting from the search step. We observe FPS is consistently superior to random sampling, under different selection ratios and targets. Improvement of FPS over random sampling first increase and then decreases, with peak improvement happening at the 30-60% ratio.

Table 3.4: Generalization comparison of various training sets: those generated by random sampling, greedy sampling, and SnP, as well as the source data pool. AlicePerson is used as the target. We directly test the trained models on Market, Duke, MSMT and report the individual and the average rank-1 accuracy (%) and mAP (%).

Training set	Market		Duke		MSMT		Average		AlicePerson	
	R1	mAP	R1	mAP	R1	mAP	R1	mAP	R1	mAP
Source pool	67.36	43.16	**62.39**	**41.15**	**35.72**	**15.40**	**55.16**	**33.23**	**38.96**	17.62
Random	6.44	1.92	38.29	18.75	19.90	6.42	21.54	9.03	23.67	9.80
Greedy	74.11	**51.98**	26.08	13.27	7.32	2.13	35.83	22.46	33.22	15.10
SnP	**74.32**	51.03	27.06	13.43	8.19	2.28	36.52	22.24	38.23	**18.17**

Table 3.5: Comparison of different source pools. A: Market + Duke + MSMT. B: A + UnrealPerson. C: B + RandPerson + PersonX. All: C + CUHK03 + RAiD + VIPeR + PKU-Reid. SnP is consistently effective for different pools. 2% IDs are selected from the source pool. Notations and evaluation metrics are the same as those in the previous table.

Search method	Pool A		Pool B		Pool C		All	
	R1	mAP	R1	mAP	R1	mAP	R1	mAP
Random	11.47	4.01	21.56	8.61	23.60	9.39	23.67	9.80
Greedy	8.04	2.81	18.66	7.63	24.26	9.56	33.22	15.10
SnP	**14.17**	**4.47**	**25.44**	**10.16**	**36.32**	**17.22**	**38.23**	**18.17**

3.5.4 Further Analysis

Generalization analysis of the training set generated by SnP. In Table 3.4, we directly apply the re-ID model obtained from the AlicePerson-specific training set to various test sets, such as Market, Duke, and MSMT. Compared with the model trained on the source pool, our model trained with the searched training set with SnP has high accuracy on the target it trained on (*i.e.*, AlicePerson). However, the model trained with the searched training set has relatively low accuracy on domains other than its search target (MSMT for example shown in Table 3.4). This demonstrates the generated training set is target-specific, improving re-ID accuracy on the given domain, while its generalization ability is weakened. Still, its application is promising given its target-specificity strength. And in this chapter, our SnP framework can construct training data for any target data distribution, then generalization ability is less desired.

Analysis of different source pools. We compare various source pools when AlicerPerson is the target domain. From Table 3.5, we observe that our method gives very stable accuracy when using different source pools. In comparison, the compared methods, *i.e.*, random sampling and greedy sampling, as much less stable. For example, random sampling performs well under Pool A which only contains real-world images and might be similar to the target domain. But it is much poorer under Pool C which has more synthetic data, meaning a larger domain gap from the target.

Analysis of hyperparameter J. In Fig. 3.8, we show the increase of the number of clusters J lowers the domain gap between the target and searched training set and

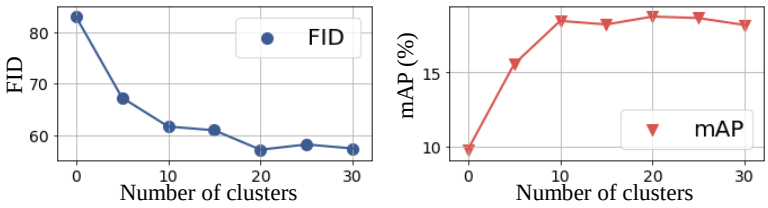

Figure 3.8: Impact of the number of clusters J to the domain gap (FID) between searched and target (left), and re-ID accuracy (right). AlicePerson is targeted, and the IDE [96] model is trained.

generally increases the task accuracy. It reaches saturation after $J = 20$. In this chapter, we use $J = 50$ in all setups.

Correlation between the number of images and training set quality. While the correlation between the number of IDs and training set quality has been shown in Fig. 3.3. We show that when the number of IDs is fixed, there is a correlation between the number of images and the rank-1 accuracy in Fig. 3.7. When we increase the number of images, it brings a significant increase in rank-1 accuracy.

3.6 Conclusion

This thesis studies the training data set search problem for object re-ID applications. Under a certain budget, we aim to find a target-specific training set that gives a competitive re-ID model. We show our method is overall superior to existing strategies such as random sampling and greedy sampling in terms of accuracy on the target domain. We analyze various components in the SnP system and find them to be stable under various source pools and targets. We also point out the correlation between domain gap, dataset size, and training set quality, and would like to further study the data-centric problems in the community.

3.7 Source Pool Details

We show the composition (tree map) of the source pool for the training set search in Fig. 3.9. On the left of Fig. 3.9, for person re-ID, we summarize the details of each dataset in rectangles and show some examples. The size of each rectangle refers to the number of images in the dataset, while the color of the rectangle indicates the domain gap of that dataset to the target set. Regarding the target set in person re-ID, we use AlicePerson [52] for calculating the FID.

We summarize the details of vehicle re-ID datasets in the right of Fig. 3.9. The size and color of each rectangle carry the same meanings as those on the left of Fig. 3.9. The AliceVehicle [52] is used as the target set for calculating FID.

The datasets we use for the source pool are standard benchmarks, which are publicly available. We list their open-source as follows. For person re-ID:

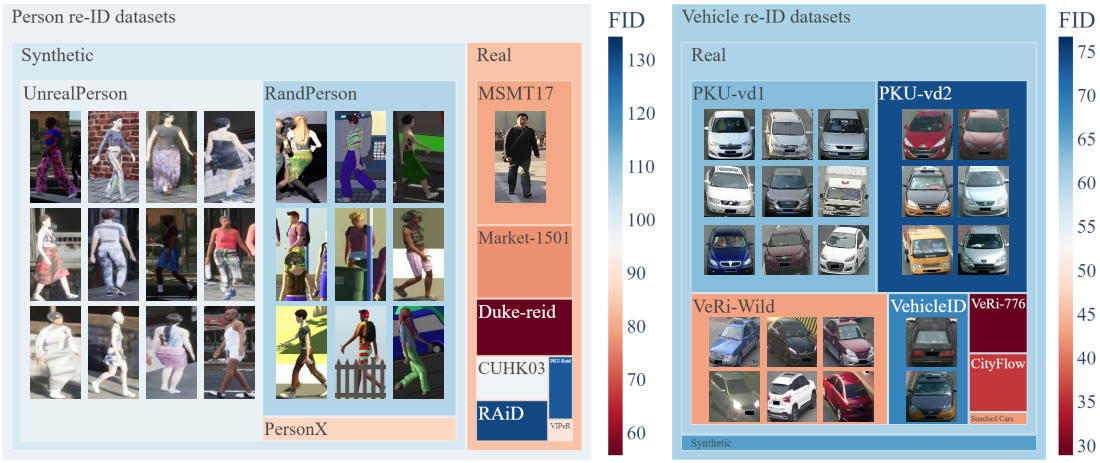

Figure 3.9: The composition (tree map) of the source pool for the training set search. (**Left**:) Person re-ID datasets were collected for constructing the source pool. (**Right**:) Vehicle re-ID datasets. For both person re-ID and vehicle re-ID, we use synthetic data and real data. Each innermost rectangle represents a publicly available object re-ID dataset. The relative size of rectangles denotes the relative image size of datasets. The relative color shows the domain gap (measured in FID) value to the target test set. In person re-ID, the target for calculating the domain gap is AlicePerson [52]. In vehicle re-ID, the target for calculating the domain gap is AliceVehicle [52].

Market [53] https://zheng-lab.cecs.anu.edu.au/Project/project_reid.html;
Duke [54] https://exposing.ai/duke_mtmc/;
MSMT17 [55] http://www.pkuvmc.com/publications/msmt17.html;
CUHK03 [91] https://www.ee.cuhk.edu.hk/~xgwang/CUHK_identification.html;
RAiD [92] https://cs-people.bu.edu/dasabir/raid.php;
PersonX [1] https://github.com/sxzrt/Instructions-of-the-PersonX-dataset;
UnrealPerson [72] https://github.com/FlyHighest/UnrealPerson;
RandPerson [79] https://github.com/VideoObjectSearch/RandPerson;
PKU-Reid [93] https://github.com/charliememory/PKU-Reid-Dataset;
VIPeR [94] https://vision.soe.ucsc.edu/node/178.
For vehicle re-ID, we also list their open-source:
VeRi [82] https://github.com/JDAI-CV/VeRidataset;
CityFlow [83] https://www.aicitychallenge.org/;
VehicleID [3] https://www.pkuml.org/resources/pku-vehicleid.html;
VeRi-Wild [81] https://github.com/PKU-IMRE/VERI-Wild;
VehicleX [2] https://github.com/yorkeyao/VehicleX;
Stanford Cars [95] http://ai.stanford.edu/~jkrause/cars/car_dataset.html;
PKU-vd1 [80] https://www.pkuml.org/resources/pku-vds.html;
PKU-vd2 [80] https://www.pkuml.org/resources/pku-vds.html.

Chapter 4

Training Set Optimization from Synthetic to Real

Data synthesis that can be conveniently performed in graphic engines provides invaluable convenience and flexibility for the computer vision community [50, 102, 103, 104, 1]. A large amount 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 generally exists between synthetic data and real-world data, as discussed in chapter 2, which can substantially decrease accuracy when a model trained on synthetic data is tested with real data [21, 103]. The domain gap problem can be addressed from the **appearance level** [21] or the **content level** (Fig. 4.1). The former often translates image style by using a neural network [19], 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 [19], but also on the content level [21]. Take vehicle re-ID as an example. Images in the VehicleID dataset [3] mainly exhibit car rears and fronts, whereas the VeRi-776 dataset [82] covers more diverse viewpoints. This difference in viewpoints is considered an example of content-level discrepancy between datasets. As a result, models trained on VehicleID [3] have a significant accuracy drop when tested on VeRi-776 [82]. 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 chapter 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. 4.1 **A**, the proposed attribute descent algorithm can automatically configure simulator attributes, such as camera viewpoint, lighting direction, and object placement, so that

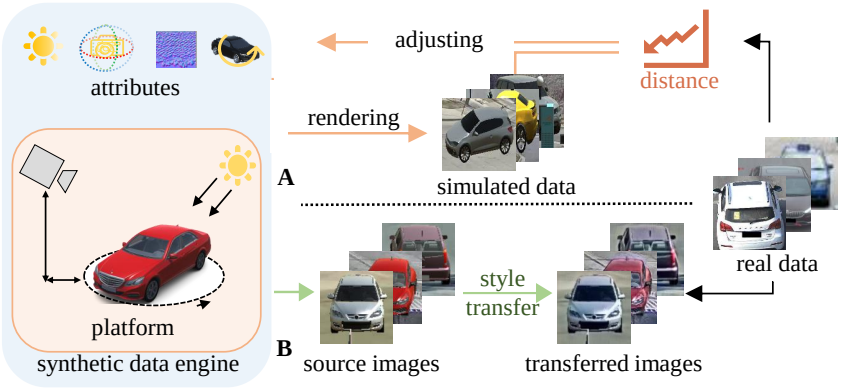

Figure 4.1: Domain adaptation on the content level (**A**) and appearance level (**B**). With real data as a target, content-level domain adaptation optimizes image content attributes such as camera viewpoint and illumination. In comparison, appearance-level and feature-level domain adaptation typically 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.

the synthesized data are close to real-world data as measured by the Fréchet Inception Distance (FID) [28]. 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 4.5.

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** and 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 chapter is an extension of our conference publication [2]. Three major differences are presented. First, while only vehicle re-ID is studied in [2], 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.

4.1 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) [105] and its variants to reduce the appearance gap [19, 106,

Table 4.1: Statistics of synthetic datasets used in this chapter and their comparison with several existing synthetic datasets. 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.

Task	Dataset	#Models	#Classes (IDs)	Attr.
Image cla.	VisDA Source [112]	1,907	12	✗
	ObjectX	1,400	7	✓
Person re-ID	RandPerson [79]	8,000	8,000	✗
	UnrealPerson [72]	3,000	3,000	✗
	PersonX [1]	1,266	1,266	✓
Vehicle re-ID	PAMTRI [76]	402	402	✓
	VehicleX	1,362	1,362	✓

36, 107, 108]. The latter models the dependence between two domains by means of feature-level statistics [109], and many moment matching schemes are studied to learn a shared feature representation [31, 34, 32, 110, 33, 111]. 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 [112], re-identification [1, 76], semantic segmentation [19, 113, 114], navigation [115] and detection [21, 116]. 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 [50, 117, 118, 119, 112, 120]. Others manually create their own simulation environments and objects [121, 113, 115, 122, 123], and find it is beneficial to use random attribute within a reasonable range to create random content [104, 124, 125]. 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 [126, 127], focusing on intrinsic properties of 3D models such as backbones, geometry and surface. The studied object models include faces [128], persons [129, 130], vehicles [127] and furniture [126]. 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* [21, 22, 114, 103]. They are closest to our work. For example, Kar *et al*. and Devaranjan *et al*. simulate traffic for training vehicle detection networks [21, 22]. Xue *et al*. and Ruiz *et al*. construct street scenes for semantic segmentation model training [114, 103]. Many of these works use

Figure 4.2: Sample 3D models for ObjectX, PersonX [1], and VehicleX [2], which are used for synthesizing images for image classification, person re-ID, and vehicle re-ID, respectively.

reinforcement learning (RL) based algorithms to optimize attributes [21, 22, 114, 103]. In this chapter, 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 4.8).

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 [10, 131, 13, 11]. Another task that we consider is object re-ID [53, 132, 82], which has many robust systems proposed recently [74, 75, 76, 77]. 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 [38]. For example, many classification datasets [10, 131, 133] 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 [134]. There are some existing approaches for dataset bias visualization. For example, t-distributed stochastic neighbor embedding (t-SNE) can exhibit *feature-level* data distributions [135]. 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 [1]. For example, vehicle images in the VehicleID dataset [3] are either captured from car front or back, whereas the VeRi dataset [82] includes a much wider range of viewpoints. Beyond dataset-dataset differences, large differences also exist between cameras in a single dataset [136]. For example, a camera filming a cross road naturally has more vehicles orientation than a camera on a straight road. In this chapter, we leverage such characteristics to learn attributes for each camera and simultaneously visualize

the attribute distributions.

4.2 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 4.2.1), build a camera model in the graphic engine, define a set of editable attributes (Section 4.2.2), and acquire synthetic images by varying these attributes (Section 4.2.4).

4.2.1 3D Asset Acquisition

For object-centric tasks, we use 3D assets from three sources. Their details are provided below as well as in Table A.1, and sample assets are shown in Fig. 4.2.

We reformat **ObjectX** from ShapeNet-V2 [137] to simulate classification data. Similarly to ImageNet [10], ShapeNet organizes 3D shapes according to the WordNet hierarchy [138]. From ShapeNet, we select the classes that are also included in the VisDA target dataset (containing real-world images) [112], 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 A.1 shows the statistics of ObjectX, and Fig. 4.2 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*. [1] 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 [2], 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 [76]) is presented in Table A.1. 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

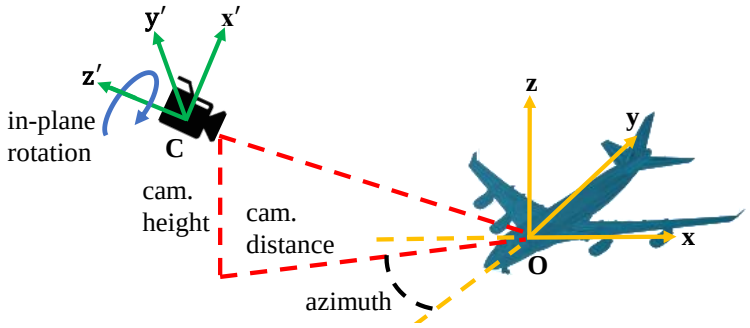

Figure 4.3: The camera model in our simulation environment. Given fixed focal length and resolution, the transformation matrix between the world coordinate system O and camera coordinate system C is determined by the azimuth, camera height, camera distance, and in-plane rotation.

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].

4.2.2 Camera Model

To capture 2D images of the 3D objects, we create a camera model, as illustrated in Fig. 4.3. 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 α 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 α, h, d and θ, 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 γ. 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

[1]https://www.aicitychallenge.org/

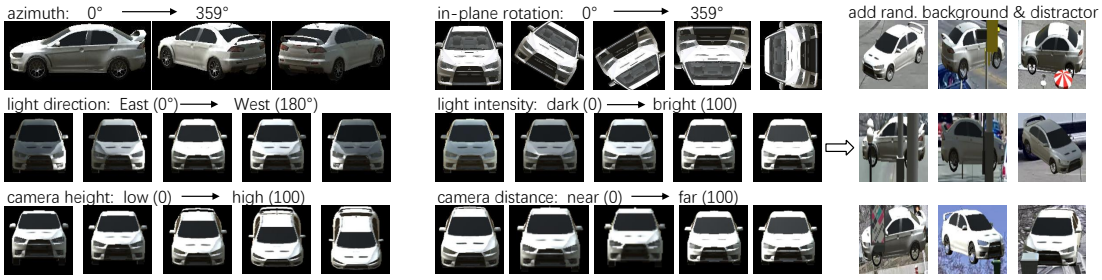

Figure 4.4: (**Left** and **Middle:**) Illustration of editable attributes. We can rotate the object (azimuth) and camera (in-plane rotation), edit light direction and intensity, and change the camera height and distance. Value and range are shown for each attribute. (**Right:**) After attribute optimization, we further add random backgrounds and occlusions to images when synthesizing the final training dataset.

projection matrix can be written as,

$$P = \underbrace{\begin{bmatrix} \gamma f & 0 & 0 \\ 0 & \gamma f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic parameters}} \underbrace{\begin{bmatrix} R(\alpha, h, d, \theta)_{3\times 3} & T(h, d)_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix}}_{\text{extrinsic parameters}}. \tag{4.1}$$

Thus, given focal length f and resolution γ, the camera model has 4 remaining attributes to be configured: azimuth α, camera height h, camera distance d and in-plane rotation θ. 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 [2]. The additionally introduced attribute "in-plane rotation" allows us to more flexibly model objects that do not usually "stand" on the ground plane.

4.2.3 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 4.2.2, 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° and 359°.
- In-plane rotation controls camera rotation on the z' axis. Its value is also between 0° and 359°.
- 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. 4.4) 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 chapter, we try to keep our investigation focused and in depth and leave the more complex applications for future work.

4.2.4 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 [139], 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 [83], 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. This strategy allows us to train models that are more robust to various backgrounds and occlusions.

[2]https://github.com/yorkeyao/VehicleX

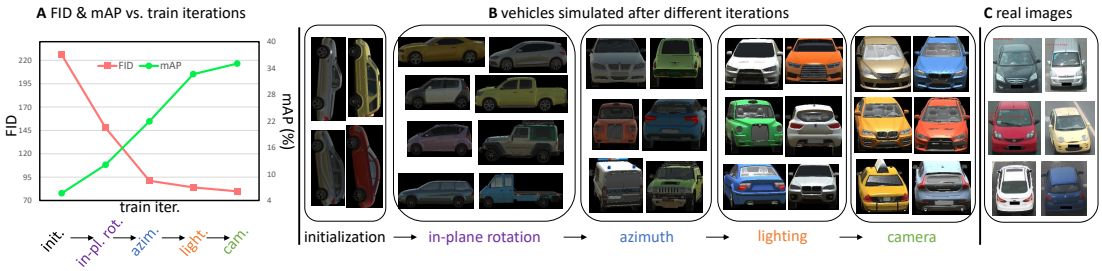

Figure 4.5: Visualization of the attribute descent process on the VehicleID [3] dataset. (**A:**) FID and task accuracy mean average precision (mAP) (%) *vs.* training iterations. We observe that the FID between synthetic data and target real data (**C**) successively decreases and that mAP on the target domain gradually increases (higher is better). "In-pl. rot." denotes in-plane rotation, "lighting" represents light direction and intensity, and "cam." means camera height and distance. (**B:**) we show the synthetic vehicles after optimizing each attribute in a certain epoch. We initialize attributes by setting the in-plane rotation to 90°, orientation to the left (0), light intensity to dark (0), light direction to east (0), camera height to the same level as the vehicle center (0), and camera distance to medium (50). Along the iterations, the content of these synthetic images becomes increasingly similar to (**C**) the target real-world images.

4.3 Proposed Method

4.3.1 Attribute Distribution Modeling

We model the distribution of the 6 attributes (Section 4.2.3) 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 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.* [140]. 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 $\boldsymbol{A} = [A_1, A_2, ..., A_K]^T \in \mathbb{R}^K$ represent attributes discussed in Section 4.2.3, where K stands for the total number of attributes. $A_i \in \mathbb{R}, i = 1, ..., K$ is the random variable representing the ith attribute. As discussed in Section 4.2.3, 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 $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_M]^T$ represent all the learnable parameters, where M is the number of these learnable parameters, and $\theta_i \in \mathbb{R}, i = 1, ..., M$, is a single learnable parameter (*i.e.*, mean, as variance is fixed) of one mixture component. Thus, the $\boldsymbol{\theta}$ can be viewed as a concatenated vector of parameters of all GMMs. Subsequently, we use $G(\boldsymbol{\theta})$ to denote the overall distribution parameterized by $\boldsymbol{\theta}$, 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 θ_1, θ_2, and θ_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 θ_4.

Furthermore, to make things easier for optimization, we let $\boldsymbol{S} = [\boldsymbol{S_1}, \boldsymbol{S_2}, ..., \boldsymbol{S_M}]^T$ represent a collection of search spaces for learnable parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_M]^T$, where $\boldsymbol{S_i} \in \mathbb{R}^{d_i}, i = 1, ..., M$, is the search space of parameter θ_i. The dimension d_i of $\boldsymbol{S_i}$ is dependant on the attribute θ_i. For example, we have θ_1, θ_2, and θ_3 are learnable parameters of attribute A_1 (in-plane rotation), and their search spaces $\boldsymbol{S_1} = \boldsymbol{S_2} = \boldsymbol{S_3} = [30l \mid l = 0, 1, \ldots, 11]^T$, which is a 12-dim vector including angles between 0° and 330° with an increment of 30°. A complete attribute distribution and search space description of all the learnable parameters is provided in Section 4.4.2.

4.3.2 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 $\boldsymbol{A}$. We further write D^s as $\{\mathcal{R}(\boldsymbol{a}_n)\}_{n=1}^N$, where N is the number of images in D^s. $\mathcal{R}(\cdot)$ is the underlying rendering function of Unity using attribute vector $\boldsymbol{a}_n$ as input and producing a synthetic image $\mathcal{R}(\boldsymbol{a}_n)$. Each attribute vector $\boldsymbol{a}_n$ is sampled from distribution parameterized by $\boldsymbol{\theta}$, *i.e.*, $\boldsymbol{a}_n \sim G(\boldsymbol{\theta})$. We aim to optimize $\boldsymbol{\theta}$ to minimize the distribution difference measure Fréchet Inception Distance (FID) [28] between D^s and D^r. Correspondingly, our objective function is written as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathrm{FID}(D^s, D^r), \tag{4.2}$$

where

$$D^s = \{\mathcal{R}(\boldsymbol{a}_n)\}_{n=1}^N, \boldsymbol{a}_n \sim G(\boldsymbol{\theta}), \tag{4.3}$$

and

$$\begin{aligned} \text{FID}(D^s, D^r) = \|\boldsymbol{\mu}^s - \boldsymbol{\mu}^r\|_2^2 + \\ Tr(\boldsymbol{\Sigma}^s + \boldsymbol{\Sigma}^r - 2(\boldsymbol{\Sigma}^s\boldsymbol{\Sigma}^r)^{\frac{1}{2}}). \end{aligned} \tag{4.4}$$

In Eq. 4.4, $\boldsymbol{\mu}^s \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}^s \in \mathbb{R}^{d\times d}$ denote the mean and covariance matrix, respectively, of the image descriptors of the synthetic dataset D^s, and $\boldsymbol{\mu}^r$ and $\boldsymbol{\Sigma}^r$ 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 [51] pre-trained on the ImageNet dataset [10].

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 [23, 21]. The latter requires training the task model (*e.g.* object detection [141]) 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 (Eq. 4.2, Eq. 4.3 and Eq. 4.4) is non-differentiable with respect to $\boldsymbol{\theta}$, 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 [142]. 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 $\boldsymbol{\theta}$ which encodes the distribution of the configurable attributes. Following coordinate descent, we propose attribute descent (Alg. 3) 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.

Formally, to iteratively optimize the objective function (Eq. 4.2) with respect to $\boldsymbol{\theta}$, we first initialize $\boldsymbol{\theta}$ at epoch 0:

$$\boldsymbol{\theta}^0 = [\theta_1^0, \cdots, \theta_M^0]^T. \tag{4.5}$$

Then at the jth epoch, we get $\boldsymbol{\theta}^j$ from $\boldsymbol{\theta}^{j-1}$ by iteratively solving the single variable optimization problems. Specifically, at ith iteration in epoch j, we optimize a single

Algorithm 3 Attribute Descent

Input: Initialized learnable parameters $\boldsymbol{\theta} = \left[\theta_1^0, \cdots, \theta_M^0\right]^T$, search space $\boldsymbol{S} = [\boldsymbol{S_1}, \boldsymbol{S_2}, ..., \boldsymbol{S_M}]^T$, rendering function $\mathcal{R}(\cdot)$ and target dataset D^r.
Hyperparameteters: J epochs for attribute descent and synthetic dataset size N.
Begin:
Optimal = ∞; ▷ Initialize the optimal FID value
for $j = 1$ **to** J **do** ▷ Number of epoch J
  for $i = 1$ **to** M **do** ▷ Enumerate parameters
    for $z \in \boldsymbol{S}_i$ **do** ▷ Traverse search space
      $D^s = \{\mathcal{R}(\boldsymbol{a}_n)|\boldsymbol{a}_n \sim G([\theta_1^j, \cdots, \theta_{i-1}^j,$
        $z, \theta_{i+1}^{j-1}, \cdots, \theta_M^{j-1}]^T)\}_{n=1}^N$.
      ▷ Generate synthetic dataset
      Score = FID(D^s, D^r) ▷ FID calculation
      if Score $<$ Optimal **then**
        Optimal = Score ▷ Update FID
        $\theta_i^j = z$ ▷ Update parameters
      end if
    end for
  end for
end for

parameter θ_i^j in $\boldsymbol{\theta}^j$, to its best value z in the search space $\boldsymbol{S_i}$:

$$\theta_i^j = \arg\min_{z \in \boldsymbol{S_i}} \text{FID}(D^s, D^r), \tag{4.6}$$

where

$$\begin{aligned} D^s = \{\mathcal{R}(\boldsymbol{a}_n)|\boldsymbol{a}_n \sim G([\theta_1^j, \cdots, \theta_{i-1}^j, \\ z, \theta_{i+1}^{j-1}, \cdots, \theta_M^{j-1}]^T)\}_{n=1}^N. \end{aligned} \tag{4.7}$$

In Eq. 4.6 and Eq. 4.7, an iteration is defined as the duration for which a single parameter $\theta_i^j, i = 1, ..., M$ undergoes an optimization process (from Step 7 to Step 14 in Alg. 3). 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 $\boldsymbol{\theta}$. 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 4.3, we can update each attribute by using discrete values within a range. For example, the search space for azimuth is

Figure 4.6: 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 [143] 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.

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. 4.5, 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. 4.5 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 train-

ing data quality. In addition, in Fig. 4.5 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.

4.3.3 Application Scenarios

4.3.3.1 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 4.3.1, attribute descent requires known class or camera labels, because we observe different attribute distributions for different categories or cameras; examples are shown in Section 4.4.4.

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 4.2 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 Alg. 3. 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 [144].

4.3.3.2 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 [145] 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 [145] 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 [145].

4.3.3.3 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 4.4.4). We can also use histograms to present the distribution of lighting intensity of a dataset.

4.4 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 4.4.1), we use attribute descent to synthesize a training set that has similar attribute distributions.

4.4.1 Source and Target Datasets

Image classification. We use ObjectX (described in Section 4.2.1) as the source and the VisDA [112] 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 [137] 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) [53] and DukeMTMC-reID (denoted as Duke) [132]. Market has 1,501 IDs, 12,936 training images and 19,732 gallery images filmed by 6 cameras. A totoal 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 [3] 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 exists: "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

Table 4.2: Comparing various training sets in object classification under synthetic training and data augmentation. 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.

Appli.	Training data	Type	Val-test split	Model	plane	bus	car	knife	mcycl	sktbrd	train	per-class
Syn. Tra.	VisDA Source	S	1:7	ResNet-50	72.13	38.57	72.40	5.20	90.41	28.67	84.51	56.0
	OX (Ran. Attr.)	S	1:7	ResNet-50	77.96	48.06	62.33	40.88	92.11	57.99	75.63	65.0
				DAN [34]	81.54	47.40	64.98	53.33	80.16	63.36	79.38	67.2
				ADDA [146]	83.01	51.01	60.11	55.65	69.63	71.03	71.91	66.0
				SHOT [147]	88.43	64.98	67.24	66.23	85.35	78.05	75.66	72.4
	OX (Attr. Desc.)	S	1:7	ResNet50	83.82	68.88	62.20	64.90	86.57	38.65	92.96	71.1
				DAN [34]	86.93	67.68	69.02	46.50	84.50	56.39	92.66	72.0
				ADDA [146]	91.88	69.00	70.26	34.38	83.67	64.81	90.69	72.1
				SHOT [147]	93.92	78.26	75.09	42.20	91.13	78.35	88.83	78.3
Data Aug.	VisDA Tar.	R	1:7	ResNet-50	96.24	85.69	91.47	94.33	92.61	90.88	90.31	91.6
	VisDA Tar.+OX (Ran. Attr.)	R+S			94.89	87.18	93.55	93.39	93.77	91.03	88.69	91.8
	VisDA Tar.+OX (Attr. Desc.)	R+S			96.90	89.86	94.33	95.04	94.28	92.38	92.07	93.6
	VisDA Tar.	R	1:1	ResNet-50	98.13	89.38	95.96	96.82	95.34	95.44	93.39	94.9
	VisDA Tar.+OX (Ran. Attr.)	R+S			96.02	86.74	94.24	93.83	93.47	90.53	89.07	92.0
	VisDA Tar.+OX (Attr. Desc.)	R+S			98.08	91.56	95.90	96.24	96.00	96.00	94.48	95.5

vehicles and 20,038 images. The VeRi-776 dataset [82] contains 49,357 images of 776 vehicles captured by 20 cameras. The vehicle viewpoints and illumination cover a diverse range. The training set has 37,778 images, corresponding to 576 identities; the test set has 11,579 images of 200 identities. There are 1,678 query images. The train / test sets share the same 20 cameras. We use "VeRi" for short in what follows. CityFlow-reID [83] 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.

4.4.2 Experimental Details

Attribute descent settings. As discussed in Section 4.3.1, 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 4.3.1, only means of the Gaussians are optimized, and initialized from the lowest value in the search space. For each learnable mean value $\theta_i, i = 1, ..., M$ in $\boldsymbol{\theta}$, the search space is specified in the range defined in Section 4.2.3,

Table 4.3: Comparison of various training sets in person reID under synthetic training and data augmentation. 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 4.2. mAP (%) and CMC scores (%) are reported.

Appl.	Training data	Type	Model	Rank-1	Rank-5	mAP
Syn. Training	ImageNet	R	IDE [96]	6.38	14.55	1.92
	Duke			42.31	61.88	18.07
	MSMT			41.98	61.67	20.46
	PX (Ran. Attr.)	S	IDE [96]	17.01	33.49	6.30
	PX (Attr. Desc.)			34.71	51.60	15.01
Data Aug.	Market	R	IDE [96]	85.30	93.82	67.84
			PCB [97]	92.49	96.85	76.67
			CBN [148]	94.35	97.91	83.63
			TransReid [98]	94.72	98.47	88.03
	Mar.+PX (Ran. Attr.)	R+S	IDE [96]	84.62	94.30	67.56
	Mar.+PX (Attr. Desc.)		IDE [96]	87.23	94.60	71.17
	Mar.+PX (Attr. Desc.)		PCB [97]	92.58	97.24	79.99
	Mar.+PX (Attr. Desc.)		TransReid [98]	**95.24**	**98.57**	**88.76**

Table 4.4: Comparison of various training set when using **Duke** as the target domain. Other setting and evaluation metrics are identical to Table 4.3.

Appl.	Training data	Type	Model	Rank-1	Rank-5	mAP
Syn. Training	ImageNet	R	IDE [96]	4.76	11.09	1.63
	Market			32.63	47.94	17.39
	MSMT			46.50	64.59	28.04
	PX (Ran. Attr.)	S	IDE [96]	22.17	38.96	10.09
	PX (Attr. Desc.)			30.83	47.80	15.91
Data Aug.	Duke	R	IDE [96]	78.14	88.29	58.93
			PCB [97]	82.99	90.53	67.47
			CBN [148]	84.82	92.51	70.13
			TransReid [98]	89.90	95.74	81.23
	Duke+PX (Ran. Attr.)	R+S	IDE [96]	75.99	87.70	55.77
	Duke+PX (Attr. Desc.)		IDE [96]	78.28	89.27	59.10
	Duke+PX (Attr. Desc.)		PCB [97]	84.25	92.15	70.71
	Duke+PX (Attr. Desc.)		TransReid [98]	**90.41**	**96.03**	**81.43**

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 pre-defined 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. After obtaining content-adapted data, we apply appearance level style transformation. For object re-ID (both person and vehicle), we use SPGAN [36] 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 [13] to classify the 7 classes. For person re-ID, we use multiple task models, including ID-discriminative embedding (IDE) [96], the part-based convolution (PCB) [97], and TransReid [98]. Note that when implementing these task models, we use their official

Table 4.5: Comparison of various training sets when VehicleX is the source domain and **VehicleID** is the target domain. We examine the use case of training data augmentation. Data type notations and evaluation metrics are the same as those in Table 4.3. "Small", "Medium" and "Large" refer to the three test splits of the VehicleID test set [3]. "VX" denotes the VehicleX dataset. "VID" means the VehicleID dataset.

Training data	Type	Model	Small			Medium			Large		
			Rank-1	Rank-5	mAP	Rank-1	Rank-5	mAP	Rank-1	Rank-5	mAP
VID	R	RAM [149]	75.2	91.5	-	72.3	87.0	-	67.7	84.5	-
		AAVER [74]	74.69	93.82	-	68.62	89.95	-	63.54	85.64	-
		GSTE [150]	75.9	84.2	75.4	74.8	83.6	74.3	74.0	82.7	72.4
VID	R	IDE [96]	77.35	90.28	83.10	75.24	87.45	80.73	72.78	85.56	78.51
VID+VX (Ran. Attr.)	R+S		80.2	93.98	85.95	76.94	90.84	82.67	73.45	88.66	80.55
VID+VX (Attr. Desc.)	R+S		**81.50**	**94.85**	**87.33**	**77.62**	**92.20**	**83.88**	**74.87**	**89.90**	**81.35**

Table 4.6: Comparison of various training sets when **VeRi** is the target domain. Data type annotations and evaluation metrics are the same as those in previous tables. The IDE and PCB baseline models are evaluated.

Appl.	Training data	Type	Model	Rank-1	Rank-5	mAP
Syn. Training	ImageNet	R	IDE	30.57	47.85	8.19
	VehicleID [3]			59.24	71.16	20.32
	Cityflow [83]			69.96	81.35	26.71
	VX (Ran. Attr.)	S	IDE [96]	43.56	61.98	18.36
	VX (Attr. Desc.)			51.25	67.70	21.29
Data Aug.	VeRi	R	IDE [96]	92.73	95.99	66.54
			VANet [151]	89.78	95.99	66.34
			AAVER [74]	90.17	94.34	66.35
			PCB [97]	94.04	98.21	72.04
	VeRi+PAMTRI [76]	R+S	PAMTRI [76]	92.86	96.97	71.88
	VeRi+VX (Ran. Attr.)		IDE [96]	93.21	96.20	69.28
	VeRi+VX (Attr. Desc.)		IDE [96]	93.44	97.26	70.62
	VeRi+VX (Attr. Desc.)		PCB [97]	**94.34**	**97.91**	**74.51**

implementations with default hyperparameters including learning rate and training epochs. For IDE, we adopt the strategy from [99] which uses ResNet-50 [13] 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.

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

Effectiveness of attribute descent for synthetic-data-only training. After gener-

Table 4.7: Comparison of various training sets when **CityFlow** is the target domain. Data type annotations and evaluation metrics are the same as those in previous tables. We use IDE [96] as the task model with both the cross-entropy (CE) loss and triplet loss.

Training data	Type	Model	Rank-1	Rank-20	mAP
CityFlow	R	BA	49.62	80.04	25.61
		BS	49.05	78.80	25.57
		IDE (CE+Tri.)	56.75	72.24	30.21
CityFlow+PAMTRI [76]	R+S	PAMTRI [76]	59.7	80.13	33.81
CityFlow+VX (Ran. Attr.)		IDE (CE+Tri.) [96]	63.59	82.60	35.96
CityFlow+VX (Attr. Desc.)		IDE (CE+Tri.) [96]	**64.07**	**83.27**	**37.16**

ating content-adapted synthetic data with attribute descent (visual examples in Fig. 4.6), 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 [104, 23]. 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 Table 4.2, Table 4.3, Table 4.4, Table 4.8, Table 4.6 and Table 4.7, 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 +6.1% 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. 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 4.3.3.2), 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 Table 4.2, Table 4.3, Table 4.4, Table 4.5, Table 4.6 and Table 4.7. 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 *appearance*

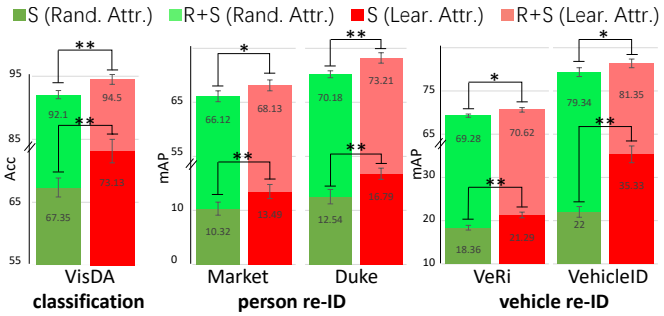

Figure 4.7: Comparison of training sets synthesized from learned attributes and random attributes. Experiments are conducted on image classification (VisDA), person re-ID (Market and Duke), and vehicle re-ID (VeRi and VehicleID). Top-1 recognition accuracy (%) and mAP (%) are used. Two application scenarios are evaluated: training with synthetic data only ("S") and training data augmentation ("R+S"). We clearly observe that learned attributes contribute to better training sets. Statistical significance analysis is performed, where $*$ means statistically significant (*i.e.*, $0.01 < p$-value < 0.05) and $**$ denotes statistically very significant (*i.e.*, p-value < 0.01).

discrepancy between source the target data. To summarize, the superiority of attribute descent over random attributes is also shown in Fig. 4.7.

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 4.2 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 4.3, 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 4.4) and the VehicleX to VeRi setting (Table 4.6).

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 [4], and only reinforcement learning has been previously used for content-level domain adaptation [103].

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 [152]. For Bayesian optimization, we use the pipeline stated in [153]. For reinforcement learning, we reproduce the LTS structure [103] 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

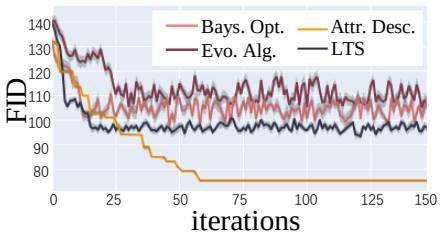

Figure 4.8: Convergence comparison between attribute descent and existing gradient-free methods including Bayesian optimization, evolutionary algorithm and LTS. Gray regions show the error bar for loss curves.

of training (*i.e.*, computing the FID score 200 times and taking the lowest FID score). Comparison results are shown in Fig. 4.8 and Table 4.8, from which four observations are made.

First, under the same task network IDE [96], learned attributes (regardless of which optimization method is used) outperform random attributes in both FID and mAP, demonstrating the benefit of attribute learning for alleviating content differences. 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 [4]. But in our search space, all the attributes significantly contribute to the distribution differences as shown in Fig. 4.5 and Fig. 4.11. 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. 4.8, 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 chapter 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 Table 4.3, Table 4.4, Table 4.5, Table 4.6 and Table 4.7. For example, when Market is jointly trained with personX, our system outperforms TransReID [98] with real data only by +0.52% in Rank-1 accuracy. Similarly, on Duke and VehicleID (small), our system exceeds TransReID [98] and GSTE [150] 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 [76].

Impact of attribute order in attribute descent. In Fig. 4.10, 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

Table 4.8: Comparison of attribute descent and existing gradient-free methods. The training set comparison when **VehicleX** is the source domain and **VehicleID** is the target domain, under training with synthetic data only. The IDE [96] model and the "Large" train-test split is used. Lower FID indicates lower domain discrepancy with VehicleID.

Training data	Type	FID↓	Rank-1↑	Rank-5↑	mAP↑
VeRi	R	45.39	28.00	41.11	38.59
Cityflow		75.36	38.23	53.70	45.57
VX (Ran. Attr.)	S	134.75	18.76	30.11	22.00
VX (Ran. Sear.)		109.94	21.84	35.29	26.35
VX (Evo. Algo.)		105.14	21.97	35.78	27.12
VX (Bay. Opti.)		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

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 more for 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. 4.11. 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%).

4.4.4 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 Fig. 4.9 and Fig. 4.12, 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 $< 30^\circ$. The orange points indicate in-plane rotation of $> 30^\circ$.

Viewpoint distribution and bias for various classes in VisDA. In Fig. 4.9 (right)

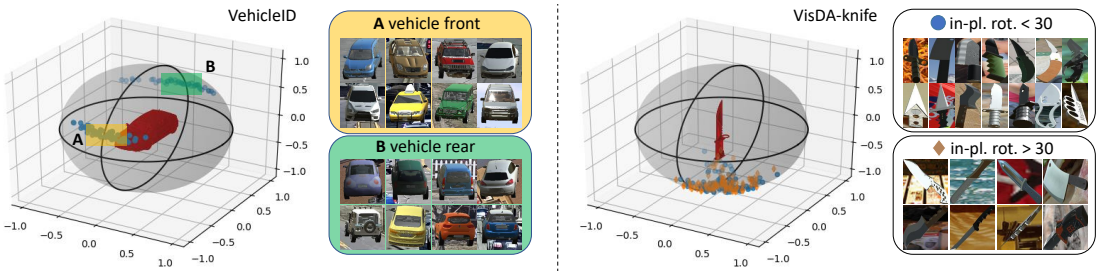

Figure 4.9: Viewpoint distribution visualization for VehicleID and class *knife* in VisDA. (**Left**:) by estimating the orientation parameters for each vehicle, our method shows that the viewpoint distribution on VehicleID is bi-modal, where orientations are concentrated in the **A** front or **B** 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 $< 30^{\circ}$ while the orange dots mean in-plane rotation $> 30^{\circ}$.

and Fig. 4.12 **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 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. 4.12 **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.

4.5 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 [21] and reinforcement learning [103]. These methods are best applied in scenarios with a relatively large number of parameters (at least hundreds or thousands) to optimize. How-

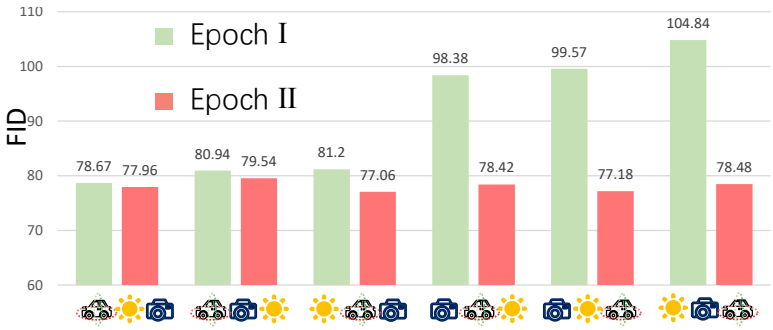

Figure 4.10: Comparison of different attribute orders in attribute descent optimization. We show FID values between generated data and the VehicleID dataset after epoch I and II in attribute descent, which is performed for a total of two epochs (see Alg. 3). Various optimization orders of attributes 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. These results suggest the correlation among attributes is weak

ever, 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 examined 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 [154, 155]. 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 [154, 155] 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 our preliminary experiment, the **discriminator** method is prone to detecting the large difference between synthetic and real data and thus continually tells the generator that its generated data have poor quality. This problem breaks the Nash equilibrium between the generator and discriminator, thus hindering us from obtaining an effective generator. On the other hand, **a task network** can provide accurate supervision signals but is infeasible when target domain labels are not provided. Furthermore, for object re-ID which is evaluated across cameras, the overall supervision provided by the task network does not reflect the training data quality in single cameras and thus poses difficulty in synthesizing data in each camera. As a result, we focus on the difference between features, using FID [28] to quantitatively measure the distribution difference between two datasets.

Application scope. To demonstrate the effectiveness of attribute descent, this chapter focuses on object-centric tasks, which are either fundamental or have very

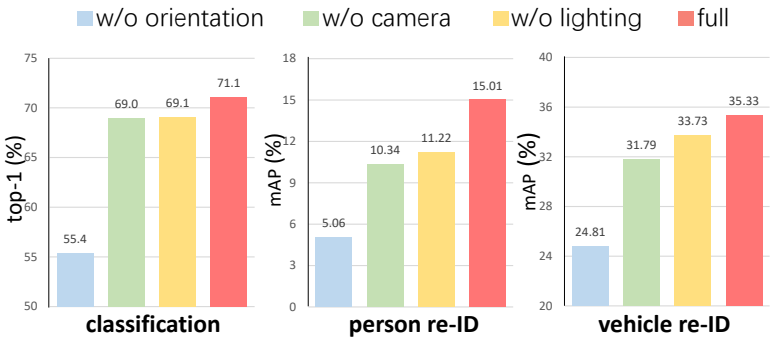

Figure 4.11: Ablation studies of each group of attributes: object orientation, camera pose and lighting. Each ablation experiment leaves a certain group of attributes unoptimized (*i.e.*, following the uniform distribution) and is compared with the full system. Top-1 accuracy (%), mAP (%) and mAP (%) are reported on image classification, person re-ID and vehicle re-ID tasks, respectively. We use VisDA, Market, and VehicleID as the target domains for the three tasks, respectively.

important applications in the real world, and a relatively small number of attributes are involved. Under these scenarios, attribute descent has quicker convergence and superior performance compared with existing gradient-free optimization methods. As such, the attribute decent serves as an effective baseline for object-centric content-level domain adaptation.

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 [114], 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 [156] as target and DeepLab-v2 segmentation model [157], we present quantitative results in Fig. 4.13. 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 [158], where we speculate that global optimization algorithms like reinforcement learning will be good alternatives. With these considerations in mind, we posit that the attribute decent serves as a straightforward and readily applicable baseline for syn2real content-level domain adaptation. Unlike the intricate hyperparameter tuning often inherent in reinforcement learning and evolutionary algorithms, the attribute descent method avoids the complexities associated with such processes. Moreover, our paper demonstrates a notable advantage of the attribute descent technique over these methods within a comparatively limited search space – specifically, within the scope of object-centric tasks as outlined in our study. This outcome underscores its promising utility as a foundational benchmark

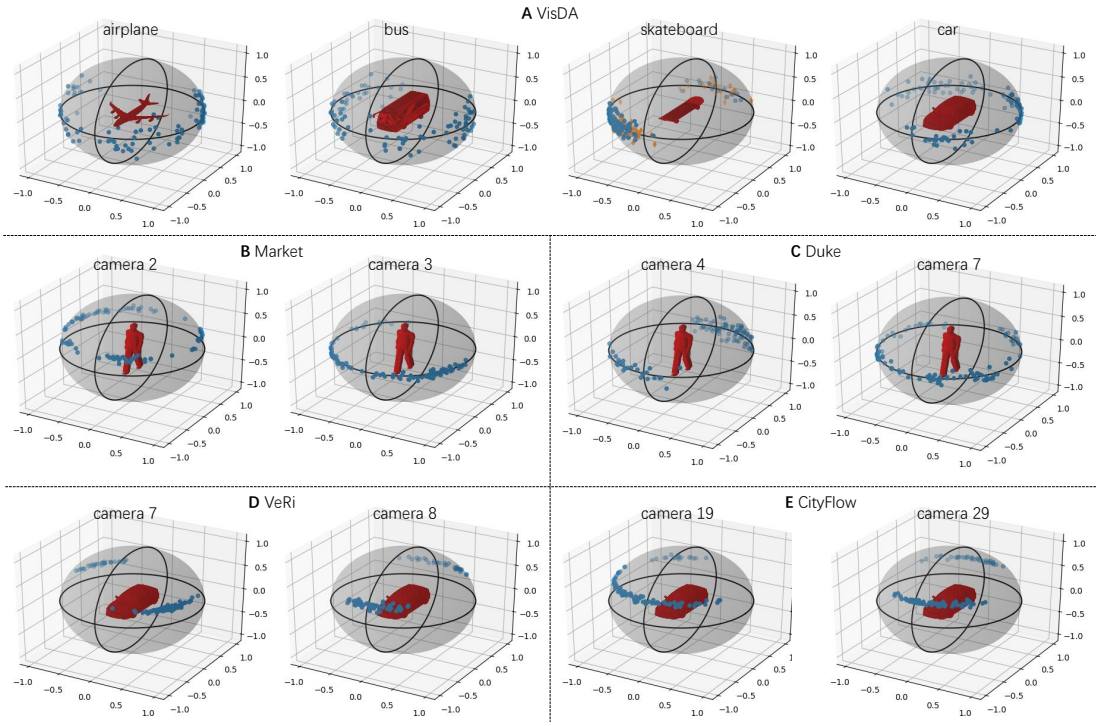

Figure 4.12: Viewpoint distribution visualization for different categories in VisDA and different cameras in Market, Duke, VeRi and CityFlow. In image classification (VisDA), we observe that each object has a specific and unique view point pattern. Likewise, each camera in the re-ID task has a distinct pattern. These patterns in classes or cameras reflect the viewpoint bias of datasets.

for syn2real content-level domain adaptation.

4.6 Conclusion

This chapter 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 graphic engine to generate training data with a 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 of given categories or cameras. This chapter demonstrates the benefit of training data engineering, and in the future, more investigations will be performed to

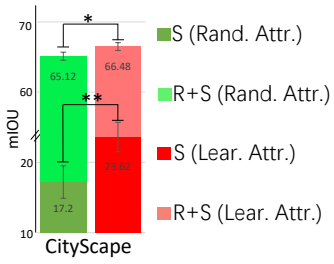

Figure 4.13: Comparison of training sets synthesized from learned attributes by attribute descent and random attributes, with application to street scene semantic segmentation. We aim to approximate the target distribution of street scenes. Similar to Fig. 4.7, two application scenarios are evaluated: training with synthetic data only ("S") and training data augmentation ("R+S"). Statistical significance analysis is conducted.

understand training data quality.

Chapter 5

Building Privacy-preserved Training Data

5.1 Introduction

The human brain is a complex system. Research towards thought patterns and expanding the way people exchange information with the outside world has never stopped for areas such as cognitive neuroscience and neurorehabilitation. With the rapid development of cognitive science, neuroscience, computer science, and signal processing technology in recent years, brain-computer interfaces (BCIs) provide human beings with other ways to communicate with the world and also allows us to get a better understanding of the physical mechanisms of human thought [159, 160].

As an essential part of brain-computer interfaces (BCIs), electroencephalograph (EEG) signals, also known as brainwaves, have found a variety of exciting and useful applications for users and have become increasingly important in various areas. Gathered from the scalp, the EEG is a signal containing information about the electrical activity of the brain. Electrodes placed on the scalp are used to detect electrical information from the brain under the scalp, bone, and other tissues. Since it is an overall measurement of the human brain's electrical activity, it contains a wealth of information. This is the reason why EEG can be applied to diverse areas like personal recognition [161], disease identification [162], sleep stage classification [163], visual image generation using brainwaves [164], and brain typing [165].

From the viewpoint of data analysis, automatic EEG analysis is challenging due to the inherent features of bio-signals. One source of ambiguity is the fused nature of features, which is common for most bio-signal feature learning tasks. The fusion here means that any one experimental trial of signals contains both wanted features and unwanted features for the given tasks. Also, due to the lack of macroscopic knowledge of the mechanism of EEG activity, this fused feature problem in EEG is more serious than for many other physiological signals. Besides that, brain wave analysis is challenging in the following aspects:

- **Low signal to noise ratio**: For EEG signals, being full of information also means full of noise and interference, making it very hard to extract reliable features

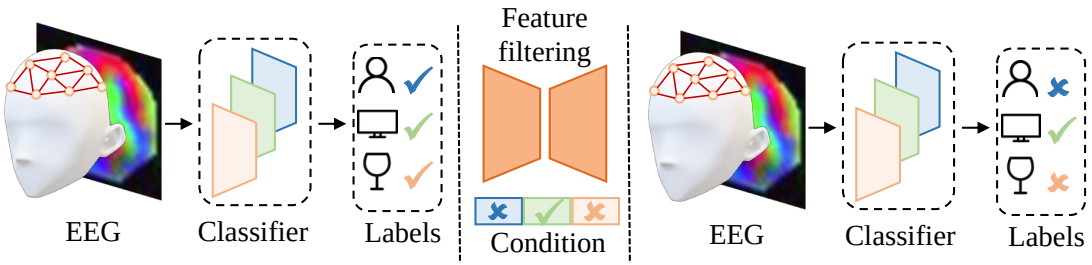

Figure 5.1: Motivation of the feature filter. We proposed the feature filter to conditionally filter out specific features in EEG signals. For example, as shown above, our model is able to filter out identities or alcoholism information with stimulus information remaining.

[166, 167, 168, 169].

- **Data format varies**: Depending on the collection device, EEG signals have a different format [170]. Hence it becomes difficult to construct standard algorithms to extract features from EEG.
- **Limited training data**: Constructing a hand-labeled training corpus for fine-grained EEG analysis is labor-intensive. Since EEG data collections are often domain-dependent, it is not practical to always collect new training data for new domains. Furthermore, due to the feature fusion problem, privacy issues are an important reason why current datasets do not involve large numbers of subjects, thereby making it practically impossible to build a huge dataset like ImageNet [171].
- **Large individual difference**: EEG signals have significant individual differences, making it hard to learn robust features across subjects [172, 173, 174].

In this work, we first designed an end-to-end framework for short-term EEG classification. To address the above difficulties, deep learning approaches are utilized in this thesis to achieve both learning and visualization. Autoencoder-based techniques are used for feature learning and dimensionality reduction for short-term EEG signals. In this thesis, this method is referred to as Image-wise autoencoders. The Image-wise autoencoders are designed based on Fast Fourier Transform (FFT) and convolutional neural networks. Using FFT, we can obtain three EEG frequency bands, then we use these frequency bands to achieve an RGB-color visualization (an image). Then, a CNN-based autoencoder is designed to extract features from these color images with both classification loss and reconstruction loss. Under this design, our models successfully overcomes the difficulties of consistent handling of EEG data.

Furthermore, in a real-world situation, customers not only require accuracy for the brain-computer interface but also require a competent level of privacy and information safety [175]. Shown in Fig. 5.1, for example, if we would like to use EEG for a personal recognition task for a bank, the only information we would like to upload is personal identity-related information. But unfortunately, EEG is a fused feature data

with a messy, vibrant symphony of personal information, including one's individuality, learning capacity, and emotion information. That is, all brain activity-related features will be uploaded and available for legal or illegal uses. For the bank example, since there currently does not exist a suitable information filtering algorithm, both the bank and potential future hackers will also be able to get our other information like disease information, emotion information, and so on. Current research has tried to specify several standards for operating on EEG data to protect users' privacy but that has not solved the problem fundamentally [176, 177, 178].

To address the above issue, for the first time, we propose a feature filter for short-term EEG signals. In practice, we do not use the idea of subtracting features to filter out properties as such properties are not well-defined. Instead, we choose to generate a new EEG trial without the unwanted features but maintaining the desired features of the original EEG trial signal. Thus, a generative adversarial network (GAN) based technique is utilized to create such an EEG signal. In this thesis, we also introduce a feature filter, which is as an extension of our short-EEG discriminative model. As mentioned earlier, the feature filter of EEG is more like a style transformation. So we are inspired by the idea of Image-to-Image translation [179] introduced in the computer vision area. This approach is designed to map one image distribution to another image distribution in order to achieve a style transformation. In this section, such a translation mechanism is used for feature filtering.

Contributions are summarized as listed:

- We propose to use the autoencoder loss to enhance the classification of EEG signals. The autoencoder loss helps to improve EEG classification accuracy.
- We propose the EEG feature filter. For the first time, we consider the situations where competent privacy information protection is generally required for customers. We transfer time series EEG signals to EEG images, thereby reducing the feature filter problem to an image translation problem.
- We conduct detailed experiments to validate the performance of the proposed network and the contribution of each component. Experiment on UCI EEG datasets shows we achieve the desired competent level of information preservation and privacy protection.

5.2 Related Works

Short-term EEG classification with deep learning. Recent advancements in EEG research have prominently featured deep learning, incorporating a range of models like Deep Belief Networks [180], Convolutional Neural Networks (CNNs) [181, 182], and Recurrent Neural Networks (RNNs) [183, 184]. These techniques are frequently used for concise EEG classification, typically acting as black-box classifiers. An example is EEGNet [172], which uses a four-layer CNN to successfully classify event-related potentials and EEG oscillations, outperforming traditional methods of extracting basic features. SyncNet [185], notable for its efficient parameter usage, achieves even better

task performance. This chapter introduces the image-wise autoencoder, an innovative method that efficiently handles noisy EEG data, leading to improved results.

The most similar work to our classification model is by Tabar and Halici [186]. They used EEG motor imagery signals and a combined CNN and fully connected stacked autoencoders (SAE) to find discriminative features. They used a Short-time Fourier transform (STFT) to build an EEG motor imagery (MI) which is unlike our 3D electrode location mapping as used in our work (described in section 3). Also, their autoencoder design is quite different from ours since they used a CNN followed by an 8-layer SAE. Nevertheless, they have demonstrated that autoencoders can help to learn robust features from EEG signals.

Privacy preserved data is garnering increasing attention in contemporary research. Previous studies in this area have primarily employed simple image transformations to safeguard privacy. This has involved the use of domain-specific knowledge and manually crafted techniques, such as pixelation, blurring, and the replacement of faces or objects, to protect sensitive information. For instance, many benchmarks manually blur human faces [52, 53] and vehicle license plates [83] for privacy reasons. A notable recent development is a shift towards automatic algorithms for creating privacy-preserved data. For example, Chamikara *et al*. propose the PEEP protocol, which applies perturbation to Eigenfaces and stores only the perturbed data in third-party servers for standard Eigenface recognition [187]. Similarly, Kiya *et al*. proposed SETR to encrypted images and then models with a vision transformer for privacy-preserving semantic segmentation [188]. Compared with existing works working on different domains, we are among the first to generate privacy-preserved EEG and use generative models for this propose.

5.3 Methodology

5.3.1 Backgrounds

An **Autoencoder** is a kind of compression algorithm, or dimension reduction algorithm, which has similar properties to Principal Components Analysis (PCA). As compared with PCA, the autoencoder has no linear constraints. The autoencoder structure has been widely used for image compression, for example [189], which inspired us to try an autoencoder based learning algorithm. An autoencoder can be divided into two parts, an encoder and a decoder. The number of nodes in the hidden layer is generally less than the nodes in the input layer and the output layer. That is, the original input is compressed to a smaller feature vector. In equation 1 below, ϕ and ψ stand for encoder and decoder, respectively, and L means squared loss. The objective of the autoencoder is to minimize the difference between the input and the generated output. A CNN based autoencoder [190] uses convolution operations as the encoder and deconvolution operations for the decoder, making it better for operating on image data. Prior to our work, a number of autoencoder-related methods have been applied to EEG signals. Stober [191] used convolutional autoencoders with custom constraints to learn features and improve generalization across subjects and trials. It achieved

commendable results but it uses CNN directly on the time domain features from EEG signals but not frequency domain features like our approach. Stober's work inspired us that it could be a general conclusion that the autoencoder-based structure can increase cross-subject accuracy, forming our basic inspiration to try autoencoder-based structures.

Generative adversarial networks (GANs) are systems of two neural networks contesting with each other in a minimax game framework [25]. The GAN approach has achieved great success in the image generation area [192, 193, 105]. GANs include two main parts, namely a generator and a discriminator. The generator is mainly used to learn the distribution of the real image and produce images in order to fool the discriminator, while the discriminator needs to accept real images while rejecting generated images. Throughout this process, the generator strives to make the generated image more realistic, while the discriminator strives to identify the real image. The key part of GAN is the adversarial loss. For the image generation task, the adversarial loss is very powerful for images in one domain transformed to the other domain since this domain cannot be discriminated by simple rules, but deep learning models have achieved some success.

The cycle-consistent adversarial network (CycleGAN) is a well-known image-to-image translation method for unpaired images [105]. It overcomes the difficulty of getting paired images and forms an autoencoder-like structure to achieve image translation. Specifically, suppose G is such a generator that generates a domain Y image from domain X, while F is the generator that generates a domain X image from the domain Y. Dx and Dy are two discriminators that are used to determine whether the coming image really belongs to domain X or domain Y, respectively. The training procedure can be separated into two symmetric parts. One is $X \rightarrow G(X) \rightarrow F(G(X))$. In this autoencoder-like loop, the training loss comes from two parts, the first is the discriminator loss which comes from Dy to judge whether G(X) is really from domain Y and the second is the reconstruction loss to judge whether F(G(X)) is the same as X or not. The other loop $Y \rightarrow F(Y) \rightarrow G(F(Y))$ works the same way.

These GAN methods are usually based on two hypotheses. One is that it is possible to build a strong classifier that can discriminate such features, and the second is the availability of a reliable generator that can filter out original features and rebuild target features. For the first hypothesis, if we cannot train a strong classifier in normal labeled training, it will be almost impossible for us to get a strong discriminator in training, because adversarial training itself is not well designed to help train the discriminator. That is not an issue for many GAN based methods which have achieved great success in the CV area, since the most popular current datasets like MNIST [194] and CIFAR-10 [46] have already achieved more than 90% accuracy using different CNNs to serve as accurate discriminators. In contrast to CV, since the NLP area does not have a universally recognized text classification method for grammar checking, current GAN methods for NLP, like Seqgan [195] and its improved version Leakgan [196] do not have a strong discriminator to guide the generator. For our second hypothesis, we have to have a strong generator that can rebuild features. However, building a strong generator is closely related to the given type of data. For the image

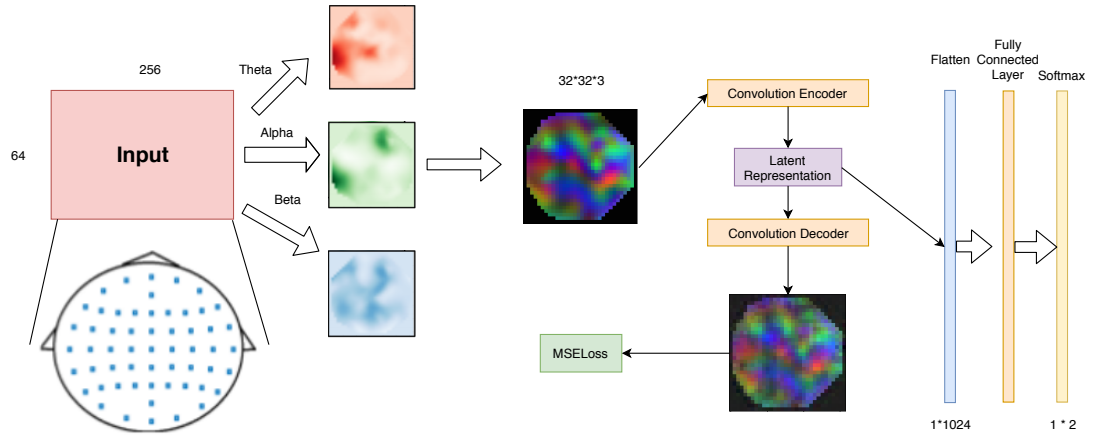

Figure 5.2: Structure of Image-wise Autoencoder. The EEG signals will be transformed to EEG images, using the geometric information of the electrode positions. When training a classifier for EEG images, the classification loss (*i.e.*, cross-entropy loss) and reconstruction loss (*i.e.*, MSE loss) are used jointly to achieve high task accuracy.

translation area, convolution and deconvolution-based methods are often used.

Image-wise autoencoders to be introduced are the solution we use to meet the two hypotheses of building a GAN for EEG. An image-wise autoencoder is used to extract discriminative and robust features from EEG images. During autoencoder training, it can reduce reconstruction loss to a very low level for the test set, making it possible to build a generator for the GAN structure. Furthermore, when we connect the features to a fully connected layer to work as a classifier, it achieves convincing results with more than 90% accuracy in the within-subject test discriminator.

5.3.2 UCI EEG Dataset

The dataset we use is from UCI, the EEG dataset from the Neurodynamics Laboratory at the State University of New York. It has a total of 122 participants with 45 control subjects and 77 subjects diagnosed with alcoholism [185, 197]. Each subject has 120 separate trial. If a subject is labeled with alcoholism, all 120 trials belonging to that subject will be labeled as alcoholism. The stimuli used are several pictures from the Snodgrass and Vanderwart picture set. It is a sort time EEG where one trial of EEG signal is of one second length. Each trial is sampled at 256Hz using 64 electrodes. For the classification task, models are first evaluated using data within subjects, which is randomly split as 7:1:2 for training, validation and testing for one person [185]. The classification objective is to discover whether the subject has been diagnosed with alcoholism or not. Also, we note that this is not a balanced dataset. It is a two-task classification but alcoholism trials account for more than 70% of the data. For training the feature filter, we also use within-subject testing but just split the source distribution (alcoholism) within subjects, which is similarly randomly split as 7:1:2 for

Table 5.1: Image-wise autoencoder structure. It contains an encoder and a decoder. The former encodes the EEG images into the representative feature, while the latter reconstructs the image from the feature.

Encoder	Decoder
Input $32 \times 32 \times 3$ Color Image	Input $16 \times 8 \times 8$ Matrix
3×3 conv, 2×2 max-pooling	3×3 deconv, 2×2 max-un-pooling
ReLU, 0.25 dropout	ReLU, 0.25 dropout
3×3 conv, 2×2 max-pooling	3×3 deconv, 2×2 max-un-pooling
ReLU, 0.25 dropout	ReLU, 0.25 dropout
3×3 conv,ReLU	3×3 deconv

training, validation and testing for each alcoholism subject. The target distribution is the whole data from control subjects. The usual challenges of handling EEG make it more difficult to apply deep learning methods compared with computer vision data or natural language processing data. The UCI EEG dataset is not an exception. First, a label is applied to one trial in this dataset. But as one trial contains 64 channels and 256 time series data, making it a 64×256 large matrix. In other words, a single EEG trial has 64×256 attributes, difficult for a neural network to find meaningful features if treated as 16,384 independent inputs. Second, EEG is a kind of time-series data but it lacks recognizable patterns in single time slices (1/sampling rate) compared with natural language processing, since each word in NLP often has a specific meaning. Third, as previous work has shown, if we consider raw EEG signals and directly use a convolution neural network for raw EEG data, there is always a serious problem to determine the size of the kernels to use at each stage [198, 186]. That is because the original features could be distributed with different time differences in a single trial depending on the scenario (different classification tasks for example), making it hard for convolution kernels to extract features.

5.3.3 Image-wise Autoencoder

The image-wise autoencoder takes images as input while using a CNN to extract features. The whole procedure is shown in Fig. 5.2, and below is some further explanation.

A. EEG to Image:

The method is derived from Bashivan's work [183], which is a method that combines the time-series information and spatial channel locations information over the scalp in a trial of EEG signals. An FFT is performed on the time series to estimate the power spectrum of the signal for each trial (64×256). From the background, we have seen theta (4-7Hz), alpha (8-13Hz), and beta (13-30Hz) waves are most representative of EEG signals when people are awake [199]. Thus, these three frequency bands are extracted from the original EEG, and the sum of squared absolute values in these frequency bands is used, forming a 64×3 map. To form an RGB EEG image, the theta frequency will be the red channel, alpha the green channel, and beta the blue channel. For each frequency band (64×1), Azimuthal Equidistant Projection (AEP) also known as Polar Projection is used to map the three-dimensional 64-

channel position into two-dimensional positions on a flat surface. That is, all EEG electrode positions are mapped into a consistent 2D space because the original EEG electrodes are distributed over the scalp in a three-dimensional fashion. In this way, each 64×1 frequency band can be mapped to a 32×32 mesh, forming $32 \times 32 \times 3$ data. The CloughTocher scheme is used for estimating the values in-between the electrodes over the 32×32 mesh. Thus, a trial of 64×256 EEG signals is transformed to $32 \times 32 \times 3$ color pictures. It is also worth noting that only frequency bands are used and the rest are omitted, thus it is likely some information is lost. However, as theta (4-7Hz), alpha (8-13Hz), and beta (13-30Hz) waves are most representative of EEG signals, to be shown in our experiment, we show that EEG images have a strong enough discriminate ability.

The motivation for this is straightforward. For the EEG2Img method, theoretically, we can adjust the size of the output EEG image as needed. For one trial of EEG signals, we can directly transfer it to one EEG image with $32 \times 32 \times 3$ format which is a very typical format in the computer vision area and there exist many mature and successful approaches and models for this format. As a result, by utilizing such a method, it is possible for us to test those models for EEG images, and leverage the computer vision community's past work on images into the EEG space.

B. Autoencoder Design:

The design of this CNN-based autoencoder is inspired by the CNN applications for CIFAR-10 dataset [46]. The CIFAR-10 dataset consists of 60,000 $32 \times 32 \times 3$ color images in 10 classes, with 6,000 images per class, with the same input dimension as our generated EEG pictures. Our encoder and decoder are described in Table 5.1. The design of the autoencoder follows Zeiler and Fergus' ideas for convolution and deconvolution [200]. The Rectified Linear Unit (ReLU) is used for activation layers to speed up the training process while dropout is performed after every activation layer to make the model more robust, since it forces all the layers before the dropout to extract redundant representations. Adam optimizer is used with 1e-4 learning rate and the batch size is set to 64. The Xavier normal initialization is used for convolution kernels.

C. Classification Task:

The features extracted from image-wise autoencoders will be flattened into a long vector, composed of 16 hidden unit representations $\times$ 64 autoencoders in the channel-wise case or $16 \times 8 \times 8$ matrix in the image-wise case. Then we use a feedforward network with three hidden layers. During the training of these three fully connected layers using a 4e-5 learning rate, the encoder of both the channel-wise and image-wise autoencoders will also be fine-tuned by the classification loss using a much smaller learning rate (1e-7).

5.3.4 Feature Filter for EEG

For the feature filter, we consider the problem of supervised domain transformation, where we are given source domain distribution X with both wanted and unwanted features, labels Z for wanted features, and target domain distribution Y with wanted

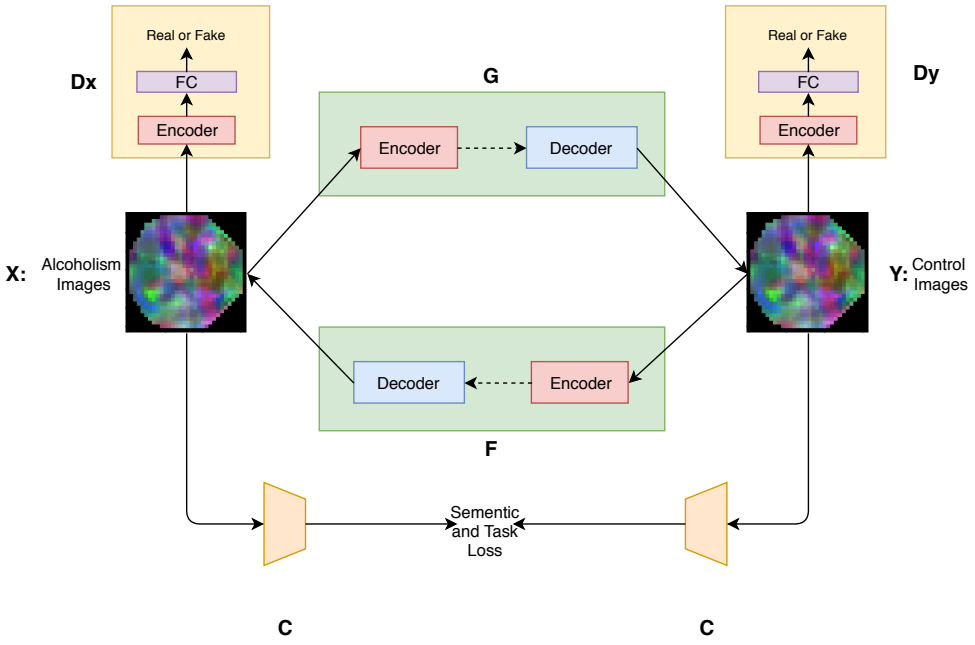

Figure 5.3: Structure of Feature Filter. Here we use the example of filtering out alcoholism information in the EEG signal. The feature filter is composed of two generators, two discriminators, and one classifier. The generators (*i.e.*, G and F) are used for distribution mapping. The discriminators (*i.e.*, D_x and D_y) are used to judge a sample whether is in-distribution. The classifier (*i.e.*, C) is used to maintain semantic information when mapping distributions.

features only. The given source domain distribution X is not paired with target domain distribution Y.

Shown in Fig. 5.3, for the UCI EEG dataset, the task of a feature filter is to replace EEG images with the alcoholism condition to an EEG image with the control condition. The objective of the feature filter is to directly learn a mapping from domain X to domain Y. So given an EEG image from X domain, the mapping representation in domain Y is our filter result. For this CycleGAN based Structure, the specific loss formulations are shown as follows.

5.3.4.1 Loss Formulation

The objective of the feature filter is composed of three parts: adversarial loss, autoencoder loss and sentiment and classification loss. They can be expressed as:

A. Adversarial Loss:

The adversarial loss is the key part for the mapping from one distribution to another. For achieving this, the adversarial discriminator is used to judge whether the image is real or fake. For the loop $X \rightarrow G(X) \rightarrow F(G(X))$, the ability to judge whether an image belongs to a certain distribution is given by the adversarial loss. For loop $X \rightarrow G(X) \rightarrow F(G(X))$, it is defined as:

Table 5.2: The conv-deconv generator structure in feature filter. The generator also contains an encoder and a decoder, which are mainly composed of convolution operations and deconvolution operations, respectively.

Encoder	Decoder
Input $32 \times 32 \times 3$ Color Image	Input $128 \times 8 \times 8$ Matrix
4×4 conv, Leaky ReLU, 4×4 conv, Leaky ReLU, 3×3 conv, Leaky ReLU, 3×3 conv, Leaky ReLU,	4×4 Deconv, Leaky ReLU, 4×4 Deconv, Leaky ReLU, Tanh

$$L_{GAN}(G, D_Y, X, Y) = E_{x \sim pdata(x)}[log[(1 - D_Y(G(x))]] + E_{y \sim pdata(y)}[logD_Y(y)].$$

This is generally the standard format of GAN loss and used to make sure the generated samples are convincing. The adversarial loss for the loop $Y \rightarrow F(Y) \rightarrow G(F(Y))$ is in a similar format.

In practice, however, the training of a GAN is quite unstable. Though the adversarial loss will force the generated image to look similar to real images, there is no guarantee for the direction of changes. To further make sure the feature filter meets our requirements, an autoencoder loss and sentiment loss are introduced as regularization terms.

B. Autoencoder Loss:

The autoencoder loss is also called reconstruction loss or cycle-consistency. It is an L1 loss that is used to keep $X \approx F(G(X))$, so that the generator will be forced to maintain features from the original image to have enough information to reconstruct the image during the backward loop. As a result, for loop $X \rightarrow G(X) \rightarrow F(G(X))$, it refers to:

$$L_{AL}(G, F) = E_{x \sim pdata(x)}[||F(G(x)) - x||_1].$$

Loop $Y \rightarrow F(Y) \rightarrow G(F(Y))$ has a similar autoencoder loss to ensure G(F(Y)) is similar to Y.

C. Sentiment and Task Loss:

The sentiment and task loss originates from Hoffman's CYCADA model for domain adaptation [19]. Hoffman's solution is to train a cycleGAN model with sentiment and task loss to generate fake target data $fake_Y$ from source data X_S, thereby forming ($fake_Y$, Z_S) data label pairs to advance the current state-of-the-art domain adaptation model.

Though the objective for domain adaptation is not directly related to our feature filter task, their proposed sentiment and task loss are useful for building a feature filter. In their proposed CYCADA model, the goal for using sentiment and task loss

Table 5.3: The effectiveness of image-wise autoencoders in EEG classification. We test their effectiveness on the task of alcoholism classification, under the setting of within-subject classification and cross-subject classification. We show that our image-wise autoencoder achieves competitive within-subject classification accuracy and state-of-the-art cross-subject classification accuracy.

Method	Within Sub. Acc↑	Cross Sub. Acc↑
rEED [201]	70.2	61.4
PSD [202]	81.6	60.5
DE [202]	82.1	62.2
EEGNet [172]	87.8	67.2
SyncNet [185]	**92.3**	**72.3**
w/o AE loss	91.5	71.2
w AE loss	**91.7**	**75.6**

is to maintain labeled information when generating $(fake_{X_T}, Z_S)$ data label pairs. This idea satisfies the property that the wanted features are maintained in our feature filter design.

The sentiment and task loss are given by an additional classifier C which gives labeled information. For the definition of task loss, it is basically a simple cross-entropy loss:

$$L_{task}(C, X, Z) = -E_{(x,z)\sim(X,Z)} \sum_{k=1}^{K} \mathbb{1}_{[k=z]} log(\sigma(C^{(k)}(x))),$$

where σ means the softmax function. In practice, the classifier will be trained on source domain X and wanted label Z. As a result, loss $L_{task}(C, X, Z)$ will be used to show that the target feature label is retained.

So classifier C works as a constraint by giving a semantic consistent loss. The semantic consistent loss will not take any explicit labeled information but focuses on the labeling consistency. The two generators will not change the labeled information when performing image translation. If we define $p(C, X) = argmax(C(X))$, the semantic consistency loss is as follows:

$$\begin{aligned} L_{sem}(G, F, X, Y, C) = & L_{task}(C, F(Y), p(C, Y)) \\ & + L_{task}(C, G(X), p(C, X)). \end{aligned}$$

In conclusion, using the full loss functions mentioned above, we add those loss functions, and we have the final objective function:

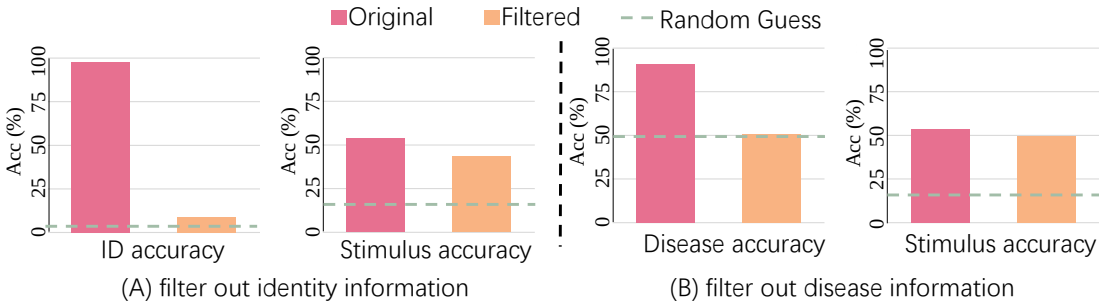

Figure 5.4: The performance of the feature filter when (A) aiming to filter out identity information and (B) to filter out disease (*i.e.*, alcoholism) information.

$$
\begin{aligned}
L_{total} = & L_{task}(C, X, Z) \\
& + L_{GAN}(G, D_Y, X, Y) + L_{GAN}(G, D_X, Y, X) \\
& + L_{AL}(G, F) + L_{AL}(F, G) \\
& + L_{sem}(G, F, X, Y, C).
\end{aligned}
$$

5.3.4.2 Network Architecture

We first use a modified version of Image-wise Autoencoder as our generator (shown in Table. 2), and our discriminator is the combination of Image-wise Autoencoder and one fully connected layer.

To improve performance, we tried the ResNet-9 generator and patchGAN combination for training. The combination of ResNet generator and patchGAN achieves the best performance in many image translation applications [105]. The residual-based generator is based on Johnson's ResNet model for super-resolution [203]. Similar to their work, our network is composed of one encoding block, nine residual blocks, and one decoding block. The encoding or decoding block uses the convolution/deconvolution InstanceNormReLU structure, and each residual block follows the residual connection structure which contains convolution-InstanceNorm-ReLU-convolution-InstanceNorm. The advantage of using ResNet-9 comes from its capability of handling deep neural networks [204], thereby making it easier for the generator to learn the mapping from the source distribution to the target distribution [13].

The patchGAN discriminator is derived from pix2pix [179], which is a paired image translation framework. The ordinary discriminator determines whether an image is real or fake from the entire image while the PatchGAN discriminator uses local patches. For loop $X \rightarrow G(X) \rightarrow F(G(X))$, The discriminator D_y takes in two images, the real image Y and the generated image G(X), pass them through 5 downsampling convolutional-BatchNorm-LeakyReLU layers, and outputs a matrix for further classification. That is, each element in the matrix corresponds to the classification of one

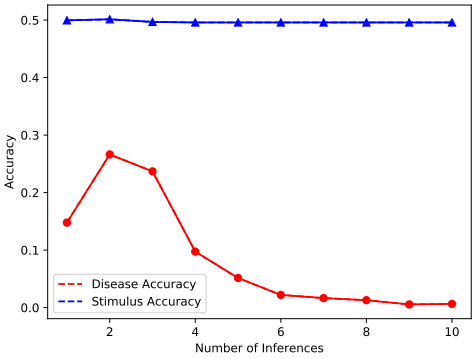

Figure 5.5: The performance of the feature filter when increasing the number of inferences, when we aim to filter out alcoholism information. We observe that the stimulus accuracy remains stable but disease accuracy essentially decreases when increasing the number of inferences.

patch. The advantage of using patchGAN is to avoid conflict with the autoencoder loss. Since we are using the final matrix to classify the image as real or fake, the patchGAN structure is used primarily to model high-frequency structure, whereas the autoencoder loss already provides low-frequency information [179].

5.4 Results and Discussion

5.4.1 Evaluation Method

The evaluation method for GAN is a difficult problem that needs to take many factors into account [205]. For a long time after the original GAN paper was published, the generated results from GANs still needed to be judged by manual selection in the CV area. The Fréchet Inception Distance (FID) and F1 scores [205] were introduced to judge the generation quality of a GAN. Both the FID and F1 scores require a strong classifier in a pre-trained in general-purposed large-scale image classification dataset (*e.g.*, ImageNet [206]), making it impossible to directly use in the bio-signal area.

Thus, we learn from the idea of using FID and Inception Score (IS) but simply use the idea of training an additional classifier to judge the classification accuracy changes. The classifier we take is still the Image-wise autoencoder with a fully connected layer (FC) which is trained separately from adversarial training. In this work, we are trying to filter out alcoholism information while keeping stimulus information. So, the desired best result should be that we get a large alcoholism accuracy reduction while keeping reasonable stimulus accuracy (low stimulus accuracy reduction) through the GAN-based autoencoder.

5.4.2 Effectiveness of Image-wise Autoencoder

The accuracy of prediction on the UCI EEG dataset, from a variety of methods, is given in Tables 4 and 5. From the result above in the within-subject test, we can see that the accuracy of our autoencoder-based method is better than most of the past methods except the SyncNet [185]. As for cross-subject results, which is the test format we are more likely to meet in real life for classifying disease, and the part we are focusing

Table 5.4: Ablation study of the feature filter structure and loss, when we aim to filter alcoholism information from EEG signals. For alcoholism accuracy, lower is the aim. For stimulus accuracy, higher is the aim.

Method	Alcoholism Acc %↓	Stimulus Acc %↑
G:Conv-Deconv D:Conv ($L_{GAN} + L_{AL}$)	18.2	47.7
G:Resnet D:PatchGAN ($L_{GAN} + L_{AL}$)	0.6	48.9
G:Resnet D:PatchGAN (L_{total})	**0.6**	**49.5**

on to improve through autoencoders. As shown in Table 5, all of our autoencoders except the shared weight channel-wise autoencoders achieve state-of-art cross-subject test accuracy. We believe this is because autoencoders can encourage feature extraction without overfitting, and will prevent the model from performing badly on new data. In other words, this prevents our model from learning the disease condition by merely remembering the personal identity and instead makes our model focus on the common features of alcoholism. This could explain why autoencoders based methods perform best in the cross-subject test. Further evidence is shown in Table 3: in order to show the performance of using an autoencoder, we construct an Image-wise CNN which has the same structure as the encoder of Image-wise autoencoder with a three-layer FC as the classifier. The result shows that though it can achieve similar within-subject accuracy as Image-wise autoencoder. it performs badly in the cross-subject test. That is, an autoencoder structure helps to improve the ability to extract robust features.

5.4.3 Effectiveness of the Feature Filter

Fig. 5.7 shows a visual example of the result of the feature filter. The left two columns map disease EEG images to control EEG images, the right two columns map control EEG images to the disease EEG images. From each direction, it can be seen that our feature filter has made a slight style transformation to images. However, those style changes are not interpretable since features from the original EEG images are not interpretable. But from Fig. 5.4 right part, initially, 90.7% of the original images are correctly classified as alcoholism. After our feature filter, only 0.6% of the images are classified as alcoholism. That is nearly all images have their alcoholism information filtered out. At the same time, stimulus accuracy has only been reduced by 4.2%, and the remaining accuracy is still well above chance since it is a 5-class classification problem.

Furthermore, one testing technique is to go through the feature filter multiple times. This idea is inspired by Ge's work for grammar error correction [207]: in their work, they observed that some sentences with multiple grammatical errors cannot be corrected by the Seq2Seq [208] inference using a single round of inference. So they involve multiple rounds of inference in both training and testing. In our work, we have not involved multiple inferences in training but merely used our trained feature filter to make multiple inferences on validation and test data. The result shown in Fig. 5.5 indicates that the result is stable after six rounds of inference. The accuracy increases

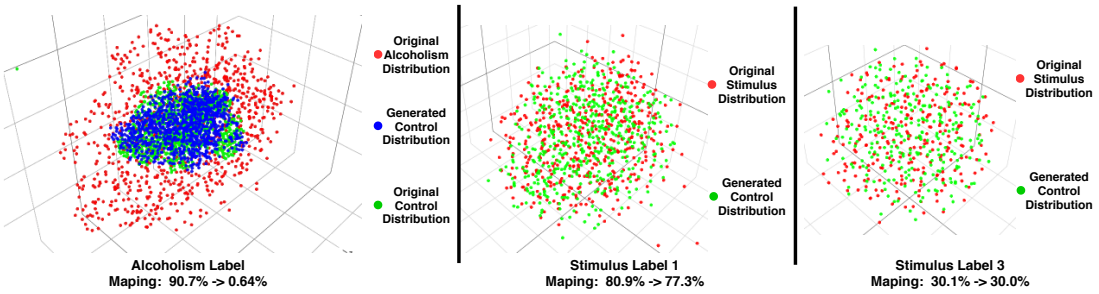

Figure 5.6: T-SNE Visualization of the feature filter output. We visualize the samples for feature filter input and output, when we aim to filter out alcoholism information. T-SNE was performed for samples with individual labeling, *i.e.*, alcoholism, stimulus label 1 and stimulus label 2.

in the first 3 rounds, we think that is because our feature filter removes unstable factors rather than filtering out the unwanted information in the first three rounds.

The performance difference between models and loss functions is shown in Table 5.4. The results show the best performance after multiple times of inferences on the test set. We can see that Resnet and patchGAN combination contribute most to the performance boost. The sentiment loss and task loss contribute to keeping stimulus information but do not achieve significant improvement in the drop of alcoholism accuracy. One hypothesis we have is that the stimulus classifier is currently far from a strong classifier. Our initial stimulus classifier at 53.7% is reasonable where chance is 20%, but cannot really be called a strong classifier. Thus, we think that could be one factor why adding semantics and task loss has not achieved a larger improvement.

5.4.4 Working Mechanism Investigation

Since style changes from EEG images are not interpretable we turn our attention to visualizing the distance between distributions. Inspired by the FID score, we can first embed our original EEG images and generated EEG images into a feature space given by some convolution layers since this feature space can be a competent representation of the original distribution. We choose to put our original EEG images and generated EEG images into the pre-trained Image-wise Autoencoder again (without the final fully connected layer) to find feature representations and then apply t-SNE visualization. Fig. 5.6 shows our t-SNE visualization results. The left part shows the mapping from the original alcoholism distribution to generated control distribution through our feature filter. We can see that the generated control image distribution is close to the original control distribution. Also, they have clear distances from the original alcoholism image distribution which matches our significant accuracy drop on alcoholism. The middle part and right part of Fig. 5.6 show the mapping from the original stimulus distribution to generated stimulus distribution for stimulus label 1 and stimulus label 3 respectively. From the middle part, originally 80.9% of data is classified correctly for stimulus data with label 1 and we still get 77.3% of data clas-

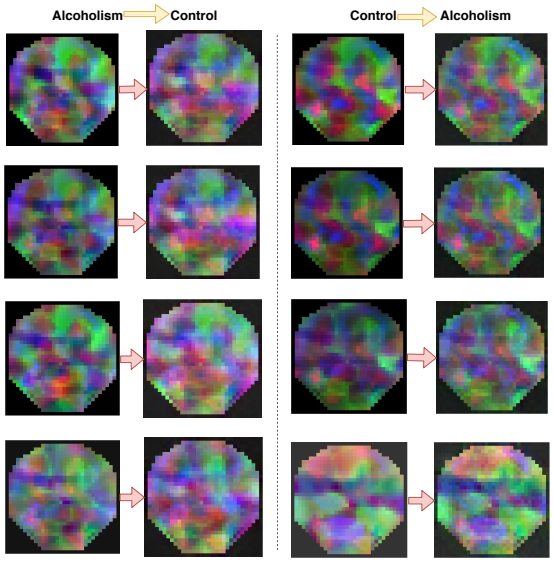

Figure 5.7: Feature filter output visualization. The example of filtering out alcoholism information in the EEG signal is shown. Noticeable changes in EEG images can be observed when feature filtering is conducted.

sified correctly after the feature filter is applied. From the right part, we can see an accuracy drop from 30.1% to 30.0% for label 3. So we can see that no matter the original classification result, the feature filter has no serious influence on the accuracy drop and t-SNE visualizations further shows that the two distributions are nearly the same though feature filters.

5.4.5 Limitation and Future Work

The first limitation is that our method is based on EEG2Img and image translation techniques, which means that it is only suitable for short-term EEG signals. The design of a feature filter for long-term EEG signals remains to be solved. We may perform frame-by-frame transformation to form a time series level solution. However, doing this may lack consistency between frames. Thus future work on this is required. The second limitation is future work for the generator; the U-net structure is also applicable for the generator since it is also the current state-of-the-art method for several image translation tasks. The third limitation is in our model: we simply stack error functions but do not really optimize the training procedure. To further reduce the loss of wanted features, we can begin with the modification of the training procedure for our GANs. A final step to increase usefulness in practice remains, in converting the filtered images to EEG signals.

5.5 Conclusion

Removing or filtering features out of EEG signals is difficult. However, building a feature filter will have a significant improvement for people's privacy protection. This

approach can lead to many useful applications. An example could be where a hospital stores only the medical condition-related EEG signal, but the bank stores only the personal identification part of an EEG (assuming a future ATM collects EEG for greater security). This chapter proposes an information-preserving feature filter, which converts the feature filtering task to an image translation task. The experimental results using accuracy drops show that our proposed feature filter can filter out nearly 90% of unwanted features and keep most of the desired features.

Chapter 6

Conclusion and Future Works

In this thesis, we have proposed several novel methods to address the problem of training set optimization, with the purpose to improve model specificity on target domains. In particular, four sub-tasks are targeted, *i.e.*, training data quality analysis, training data optimization from a source pool, training data optimization from the synthetic to real, and building privacy-preserving training data. In this final chapter, we first summarize the main content of this thesis, and then discuss the limitations of our methods which further motivate our future research directions

6.1 Conclusion

Recent years have seen significant efforts made to improve deep learning models' specificity within a specific field, with the aim of reaching performance that is on par with human capabilities. This enhancement of deep learning has been underpinned by a coalescence of the model (namely, algorithms) and data. Interestingly, the process of data optimization has been overshadowed by model optimization, receiving comparatively less attention. The objective of this thesis is to enhance the specificity of deep learning models through a focus on data. We posit that there are untapped opportunities for the improvement of the data that these models learn from, known as training data. Our approach involves the analysis of the training set quality and the creation of algorithms to optimize this training set within a variety of contexts.

We begin by assessing the quality of the training set, as its quality is integral to the success of the model built on it, and it forms the foundation of this thesis. As described in chapter 2, we have discovered that data diversity and the domain gap are two crucial elements defining the quality of training data. The former illustrates the variation among training samples, while the latter highlights the disparity in distribution between training and testing. We assert that these two elements are inherently related and should be jointly addressed. We frame the quality of training data as a function of both data diversity and domain gap, and conduct a quantitative analysis on 872 different training sets created through three methods: rotating and translating MNIST, geometric image transformations, and neural rendering. We affirm that training data of high quality is characterized by high diversity and low domain gap, and we apply this principle to the creation of superior training sets.

Next, with specific knowledge found in chapter 2 that the domain gap has a significant influence on training set quality, we examine a specific application, known as the training set search, where our goal is to generate an effective training set for a given scenario. Initially, we explore a situation where we have access to the target domain but lack the resources for real-time training data annotation, and hence, aim to construct an alternate training set from a large pool of data to yield a competitive model. In chapter 3, we suggest a search and pruning (SnP) approach to address this training data search challenge, adapted specifically for object re-identification (re-ID), an application used to match identical objects captured by different cameras. The SnP approach allows us to derive training sets that are 80% smaller than the original pool, yet still capable of achieving equal or even superior re-ID accuracy.

We also consider a training set search scenario where we can access the target domain and intend to use synthetic data to construct an alternate training set. As shown in chapter 4, to mitigate the content-level misalignment between synthetic and real data, we propose an attribute descent approach that automatically adjusts engine attributes to make synthetic data resemble real-world data. Comprehensive experiments on image classification and object re-ID confirm that the adapted synthetic data, optimized through attribute descent, can be effectively used in three different situations: training exclusively with synthetic data, training data augmentation, and numerically understanding dataset content. The scenario of training exclusively with synthetic data has data privacy and data security advantages, as it would be possible for models to be trained without any access to private/confidential data as it is only used for testing.

In addition to examining training set quality with a focus on achieving high-accuracy models, we also consider the development of a privacy-protected training set, which results in a privacy-protected model. In chapter 5, we investigated the issue of feature fusion in data and suggested a new method for removing unwanted features from data, based on our feature extractor, adapted specifically for brain-computer interface applications. These applications' input signal, electroencephalography (EEG), is known for its fusion of features and richness in various types of information from the brain. Our experimental results, obtained from an alcoholism dataset, indicate that our innovative model can filter out over 90% of alcoholism information from EEG signals on average, with a minimal loss of only 4.2% in useful feature accuracy, thereby demonstrating the effectiveness of our proposed task.

6.2 Limitations and Future Works

6.2.1 Theoretical Support for the Impact of Domain Gap and Diversity on Training Set Quality

While extensive experiments have demonstrated the combined effect of domain gap and diversity on training set quality, a more robust theoretical foundation is needed to deepen our understanding of this issue. This thesis includes an analysis of mainstream methods for creating training sets. However, it is important to note that our

review is not exhaustive, and some methods like diffusion model based methods [209] are not covered, which is included in our future work.

6.2.2 Applying Diversity in the Training Set Optimzation

In chapter 2, we posited that the domain gap and diversity together determine the quality of the training set. In practical applications, however, training set optimization may be primarily guided by the domain gap, such as Fréchet Inception Distance (FID), while diversity is often overlooked. We assume that the domain gap, under simplified conditions, has a more pronounced effect on training set quality.

Consider, for instance, as shown in chapter 3, when searching for training sets within the source pool for re-identification (re-ID) tasks. The key challenge here is the significant domain gap arising from class differences or identities (IDs). Similarly, when synthetic data is used for training, shown in chapter 4, the domain gap between synthetic and real data is considerable. In these scenarios, the magnitude of the domain gap tends to be the dominant factor affecting the training set quality.

There are instances where the original domain gap between source training and target validation is relatively small. For example, Liu *et al*. utilized "Variance Diversity", similar to our diversity metric, to guide the search of data augmentation policies involving geometric image transformations [210]. We postulate that in such cases, diversity could have a more substantial impact on training set quality. This is particularly true when geometric image transformations are used for data augmentation: here, a generally recognized in-distribution training set is typically used – that is, a training set with a minimal domain gap to the target. Under these circumstances, the goal shifts from reducing the domain gap to enhancing diversity.

Furthermore, within the context of neural rendering, and data augmentation, we experimentally found diversity and domain gap have significant influences on training set quality. It is also worth noting that it is likely to are other factors that may also have a significant influence on the training set quality when other training set creation methods are used.

6.2.3 Generalization Ability of Proposed Methods

In this thesis, we are among the first to propose the idea of training set optimization and design preliminary methods in simplified settings. However, due to the complexity of real-world scenes, though significant progress has been made in these simplified settings, it is worthwhile to test and improve the generalization ability of the proposed methods. To improve such generalization ability, we intend to work on approaches such as the following in the future:

Utilizing search and pruning frameworks in tasks other than re-identification (re-ID). Our search and pruning framework can be extended to other tasks, such as classification. However, implementing this extension would involve modifying the SnP algorithm to some extent. This is because the primary domain gap in object re-ID arises from differences in classes or identities. In contrast, classification tasks do not

exhibit such a gap since the train/val/test datasets always consist of the same classes. Consequently, our current approach of selecting similar identities based on similar distributions cannot be directly applied and must be adjusted accordingly.

Applying attribute descent on more complex synthetic environments. As delineated in Chapter 4, our focus is on testing attribute descent in object-centric applications where a single object encompasses a significant portion of an image. Future endeavors might involve examining the application of attribute descent in more intricate scenarios, such as object detection and semantic segmentation tasks.

A preliminary experiment was conducted by Xue *et al*., who incorporated attribute descent in the generation of synthetic semantic segmentation training data [114]. Their findings aligned with ours, exhibiting a substantial improvement in comparison with random attributes.

Despite the groundbreaking results from [114], their research also presents noticeable limitations. Their methodologies remain confined to road scenes, which means they are not equipped to handle arbitrary scenes. It would indeed be intriguing to see how the attribute descent algorithm can be employed for the construction of more complex synthetic scenes, aiding the creation of content-adapted training data.

As a future work talked in [211], we move forward to a more complex problem named Automated Retail Checkout (ARC), which aims to automate the retail checkout process, where users effortlessly pass products by hand through a camera, triggering automatic product detection, tracking, and counting. Compared with the object-centric tasks studied in this thesis, the synthetic film scene for ARC is significantly more difficult, including the arrangement of multiple objects instead of one object only.

Applying feature filter on tasks other than EEG classification. The implementation of our feature filter on EEG classification tasks proves to be a significant stride in privacy-preserving data processing. Having successfully maneuvered this on EEG, we may extend our feature filtering methodology to other types of data, which have their unique privacy concerns.

One of the potential fields where feature filtering can be introduced is image recognition, specifically face images. Face images carry a wealth of personal information and often form the basis for biometric identification systems. Privacy preservation in this context is essential as misuse or mishandling can lead to identity theft or unauthorized access. Applying our feature filter on face images will require an adaptation of the method. Since the features in face images are not numerical or linear as EEG signals, the filtering process will involve the recognition and extraction of essential facial characteristics while effectively obscuring personally identifiable information, and still leaving a human face visible of the appropriate age/gender/ethnicity and so on.

6.2.4 Improving deep learning models on extreme cases

In the current landscape, high-accuracy deep learning models are not uncommon. These models have shown great performance on internal benchmarks, as seen in in-

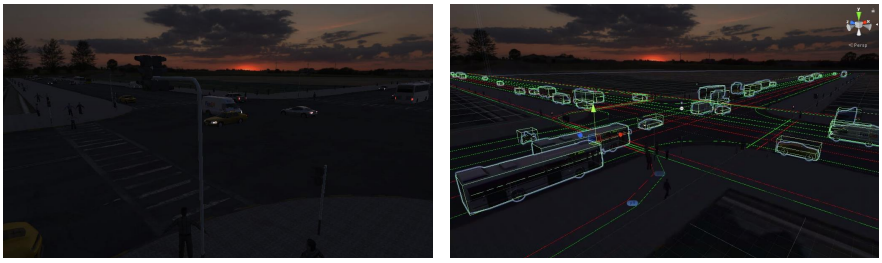

Figure 6.1: An example of a simulated road scene (left) and its automatically obtained vehicle bounding boxes (right), as an example of a corner case. In this example, the dark environment, made using VehicleX proposed in this thesis, makes vehicle detection very challenging and leads to potential detection failure. As a part of our future work, we aim to find such difficult test samples that can be used to train better AI models.

dustries such as autonomous vehicle manufacturing. However, despite their impressive accuracy, these AI systems often fail in unusual or challenging circumstances, leading to unfortunate accidents. An example could be a pedestrian hidden by an umbrella on a poorly lit day or someone positioned to cross the road in an unusual manner.

Given the critical importance of safety in these applications, it's crucial to evaluate and refine the AI system against any potential corner case scenarios. However, this task is far from straightforward. The process of gathering and annotating real-world data is costly and time-consuming, especially considering the vast number of possible scenarios.

A more cost-effective and potentially more efficient approach might be to employ 3D data synthesis engines for simulations, as mentioned in a previous chapter. These engines allow control over various elements like pedestrians, roads, weather conditions, vehicles, trees, and lighting, thereby providing a realistic yet manageable environment for the AI system. Shown in Fig. 6.1, we show our preliminary results on building such an extreme case. In this example, the dark environment makes vehicle detection very challenging and leads to detection failure. Our synthetic environment can help get such difficult samples that can be used to train better deep-learning models.

6.2.5 Exploring the intersection of content alignment, image translation, and data augmentation in syn2real

As discussed in Chapter 4, In the realm of synthetic to real, a process of content alignment and image translation, followed by data augmentation, is commonly utilized. However, as found in Chapter 2, these approaches present a paradox: while content alignment and image translation aim to minimize the domain gap, data augmentation tends to widen it. This juxtaposition leads to reduced control over the training dataset. Considering these processes from a unified perspective opens up new avenues for ex-

ploration. For instance, is there a need to integrate diversity indicators within content alignment and image translation methods? Alternatively, could the principles of data augmentation be adapted to narrow the domain gap? These questions highlight the potential for innovative strategies in optimizing machine learning methodologies.

Appendix - Dataset Contributions

To achieve our goal for training set search, we have proposed several datasets that are publicly available. We summarize them as described below.

A.1 VehicleX

We introduce a large-scale synthetic dataset generator named VehicleX that includes three components: (1) vehicles rendered using the graphics engine Unity, (2) a Python API that interacts with the Unity 3D engine, and (3) detailed labels including car type and color.

VehicleX has **a diverse range of realistic backbone models and textures**, allowing it to be able to adapt to the variance of real-world datasets. It has 272 backbones that are handcrafted by professional 3D modelers. The backbones include ten mainstream vehicle types including sedan, SUV, van, hatchback, MPV, pickup, bus, truck, estate, sportscar, and RV. Each backbone represents a real-world model. From these backbones, we obtain 1,362 variances (*i.e.,* identities) by adding various colored textures or accessories. A comparison of VehicleX with some existing vehicle re-ID datasets is presented in Table A.1. VehicleX is three times larger than the synthetic PAMTRI dataset [76] in identities, and can potentially render an unlimited number of images from various attributes. In experiments, we show that our VehicleX benefits from real-world testing either when used alone or in conjunction with a real-world training set.

In this work, VehicleX can be set to training mode and testing mode. In training mode, VehicleX will render images with a black background and these images will be used for attribute descent (see Section 4.3); in comparison, the testing mode uses random images (*e.g.,* from CityFlow [83]) as backgrounds, and generates attribute-adjusted images. In addition, to increase randomness and diversity, the testing mode contains random street objects such as lamp posts, billboards, and trash cans. Fig. 4.2 shows some sample vehicle identities.

We build the **Unity-Python interface** using the Unity ML-Agents toolkit [139]. It allows Python to modify the attributes of the environment and vehicles, and obtain the rendered images. With this API, given the attributes needed, users can easily obtain rendered images without expert knowledge about Unity. The code of this API

Table A.1: Comparison of some real-world and synthetic vehicle re-ID datasets. "Attr" denotes whether the dataset has attribute labels (*e.g.*, orientation). Our identities are different 3D models, and thus can potentially render an unlimited number of images under different environments and camera settings. VehicleX is released open source and can be used to generate (possess) an unlimited number of images (cameras).

Datasets		#IDs	#Images	#Cameras	# Attr
real	VehicleID [3]	26,328	222,629	2	✗
	CompCar [212]	4,701	136,726	-	✗
	VeRi-776 [82]	776	49,357	20	✓
	CityFlow [83]	666	56,277	40	✗
synthetic	PAMTRI [76]	402	41,000	-	✓
	VehicleX	1,362	∞	∞	✓

is released together with VehicleX.

VehicleX is a large-scale public 3D vehicle dataset, with real-world vehicle types. We focus on the vehicle re-ID task in this thesis but our proposed 3D vehicle models also have potential benefits for many other tasks, such as semantic segmentation, object detection, fine-grained classification, 3D generation, or reconstruction. It gives flexibility to computer vision systems to freely edit the content of the object, thus enabling new research in content-level image analysis. VehicleX has been used as training data in the AI city challenge in CVPR 2020 and CVPR 2021[1]

A.2 ObjectX

We have **ObjectX**, which is used for the task of image classification. It is adapted form from the ShapeNet-V2 dataset [137], which is akin to ImageNet [10], arranges 3D shapes in accordance with the WordNet hierarchy [138]. From the ShapeNet repository, we cherry-pick classes that also appear in the VisDA target dataset, comprising real-world images [112]. ObjectX contains seven classes and 200 models, chosen at random for each class. As part of our preprocessing routine, models in every class are oriented in the same direction and resized to a standard scale. Fig. 4.2 provides visual examples of 3D shapes gathered for the categories: *airplane*, *bus*, *skateboard*, *train*, *motorcycle*, and *knife*.

A.3 Automated Retail Checkout Dataset

The Automated Retail Checkout (ARC) dataset [211] is a comprehensive collection consisting of two components: synthetic images for model training purposes, and real-world data for validating and testing the model.

[1]https://www.aicitychallenge.org/

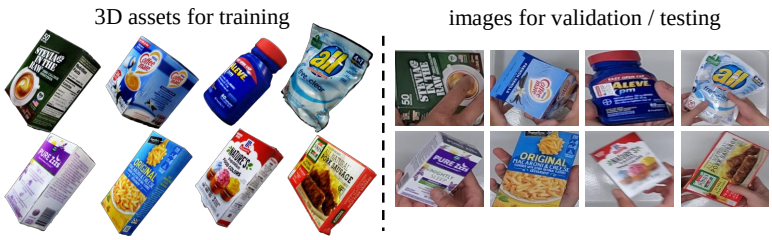

Figure A.1: The *Automated Retail Checkout (ARC)* dataset. In this dataset, we focus on the problem of using 3D assets for building 2D image classification/detection systems. Given 3D assets, we aim to construct 2D training set by setting up a filming scenario.

The synthetic dataset was generated using a camera model, as illustrated in Fig. 4.3, and elaborated upon in an earlier chapter. This process involved the collection of 116 scans of everyday retail products sourced from various supermarkets, which were then converted into 3D models. These 3D models represented a wide range of product classes, such as everyday essentials, food items, toys, furniture, household goods, among others. With these models, we were able to generate approximately 116,500 synthetic images. These images were produced in a scene represented in a Figure provided, with varied attributes like object positioning, camera angle, lighting, and backdrop used to promote diversity within the dataset. Background images were sourced from Microsoft's COCO [131], known for its diverse scene range, suitable for creating realistic image backgrounds. We have made the 3D models and Unity-Python interface accessible to participating teams, to facilitate further synthetic data creation if required.

Regarding real-world data, the test setup involved a camera fixed above a checkout counter, facing directly downwards. Customers would interact with this setup by "scanning" items at the counter as they would naturally. Multiple customers participated in this exercise, each with a unique scanning style. To help guide the AI model's focus, a shopping tray was situated under the camera. Through this exercise, we captured approximately 22 minutes of video footage, with 222 scanning events. The footage was then divided into "testA" and "testB" sets, with "testA" constituting 40% of the total footage, while "testB" comprising the remaining 60% was saved for testing and used to determine the participating teams' ranks. The ARC dataset has been employed as challenge data in the AI city challenge at CVPR 2022 and CVPR 2023.

Bibliography

[1] X. Sun and L. Zheng, "Dissecting person re-identification from the viewpoint of viewpoint," in *CVPR*, 2019.

[2] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, "Simulating content consistent vehicle datasets with attribute descent," in *Proceedings of the European Conference on Computer Vision*, 2020.

[3] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *CVPR*, 2016.

[4] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.

[5] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[6] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[8] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[9] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, W. G. Rojas, S. Diamos, G. Diamos, L. He, D. Kiela, D. Jurado *et al.*, "Dataperf: Benchmarks for data-centric ai development," *arXiv preprint arXiv:2207.10062*, 2022.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

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

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[14] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *CVPR*, 2019.

[15] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *ICML*. PMLR, 2019, pp. 2731–2741.

[16] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *NeurIPS*, 2019, pp. 6665–6675.

[17] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[18] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation." in *AAAI*, 2020.

[19] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *ICML*, 2018.

[20] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, "Simulating content consistent vehicle datasets with attribute descent," in *ECCV*, 2020.

[21] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-sim: Learning to generate synthetic datasets," in *ICCV*, 2019.

[22] J. Devaranjan, A. Kar, and S. Fidler, "Meta-sim2: Unsupervised learning of scene structure for synthetic data generation," in *ECCV*, 2020.

[23] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017.

[24] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT Press, 2016, vol. 1.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014, pp. 2672–2680.

[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *NeurIPS*, 2016, pp. 2234–2242.

[27] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," in *ICLR*, 2018.

[28] 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 *NeurIPS*, 2017.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[30] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018, pp. 586–595.

[31] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, 2016, pp. 2058–2065.

[32] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[33] Z. Zhang, M. Wang, Y. Huang, and A. Nehorai, "Aligning infinite-dimensional covariance matrices in reproducing kernel hilbert spaces for domain adaptation," in *CVPR*, 2018, pp. 3437–3445.

[34] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *ICML*. PMLR, 2015, pp. 97–105.

[35] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *NeurIPS*, 2007, pp. 513–520.

[36] 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 *CVPR*, 2018.

[37] K. Saito, K. Saenko, and M.-Y. Liu, "Coco-funit: Few-shot unsupervised image translation with a content conditioned style encoder," in *ECCV*, 2020.

[38] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR*. IEEE, 2011, pp. 1521–1528.

[39] J. Choi, C. Gao, J. C. Messou, and J.-B. Huang, "Why can't i dance in the mall? learning to mitigate scene bias in action recognition," in *NeurIPS*, 2019, pp. 853–865.

[40] M. Tian, S. Yi, H. Li, S. Li, X. Zhang, J. Shi, J. Yan, and X. Wang, "Eliminating background-bias for robust person re-identification," in *CVPR*, 2018, pp. 5794–5803.

[41] N. McLaughlin, J. M. Del Rincon, and P. Miller, "Data-augmentation for reducing dataset bias in person re-identification," in *2015 12th IEEE International conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2015, pp. 1–6.

[42] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[43] R. G. Lopes, S. J. Smullin, E. D. Cubuk, and E. Dyer, "Tradeoffs in data augmentation: An empirical study." in *ICLR*, 2021.

[44] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[45] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[46] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[47] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NeurIPS Workshops*, 2011.

[48] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *CVPR Workshops*, 2020, pp. 702–703.

[49] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[50] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *ECCV*, 2016.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[52] X. Sun, Y. Yao, S. Wang, H. Li, and L. Zheng, "Alice benchmarks: Connecting real world object re-identification with the synthetic," 2023.

[53] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[54] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *CVPR*, 2017, pp. 3754–3762.

[55] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer gan to bridge domain gap for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 79–88.

[56] H. Fan, L. Zheng, C. Yan, and Y. Yang, "Unsupervised person re-identification: Clustering and fine-tuning," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 4, pp. 1–18, 2018.

[57] Z. Zhong, L. Zheng, Z. Luo, S. Li, and Y. Yang, "Invariance matters: Exemplar memory for domain adaptive person re-identification," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[58] L. Song, Y. Xu, L. Zhang, B. Du, Q. Zhang, and X. Wang, "Learning from synthetic images via active pseudo-labeling," *IEEE Transactions on Image Processing*, 2020.

[59] J. Song, Y. Yang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Generalizable person re-identification by domain-invariant mapping network," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2019, pp. 719–728.

[60] C. Luo, C. Song, and Z. Zhang, "Generalizing person re-identification by camera-aware invariance learning and cross-domain mixup," in *European Conference on Computer Vision*. Springer, 2020, pp. 224–241.

[61] Y. Bai, J. Jiao, W. Ce, J. Liu, Y. Lou, X. Feng, and L.-Y. Duan, "Person30k: A dual-meta generalization network for person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2123–2132.

[62] B. Settles, "Active learning literature survey," 2009.

[63] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1183–1192.

[64] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *International Journal of Computer Vision*, vol. 113, no. 2, pp. 113–127, 2015.

[65] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sasrty, "A convex optimization framework for active learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 209–216.

[66] Y. Guo, "Active instance sampling via matrix partition," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[67] X. Yan, D. Acuna, and S. Fidler, "Neural data server: A large-scale search engine for transfer learning data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3893–3902.

[68] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proceedings of the European Conference on Computer Vision*, 2010.

[69] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the European Conference on Computer Vision*, 2010.

[70] Y. Lou, Y. Bai, J. Liu, S. Wang, and L.-Y. Duan, "Embedding adversarial learning for vehicle re-identification," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 3794–3807, 2019.

[71] Y. Yao, L. Zheng, X. Yang, M. Napthade, and T. Gedeon, "Attribute descent: Simulating object-centric datasets on the content level and beyond," *arXiv preprint arXiv:2202.14034*, 2022.

[72] T. Zhang, L. Xie, L. Wei, Z. Zhuang, Y. Zhang, B. Li, and Q. Tian, "Unrealperson: An adaptive pipeline towards costless person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 506–11 515.

[73] Y. Liu, Z. Wang, T. Gedeon, and L. Zheng, "How to synthesize a large-scale and trainable micro-expression dataset?" in *Proceedings of the European Conference on Computer Vision*, 2022.

[74] P. Khorramshahi, A. Kumar, N. Peri, S. S. Rambhatla, J.-C. Chen, and R. Chellappa, "A dual path modelwith adaptive attention for vehicle re-identification," in *ICCV*, 2019.

[75] Z. Wang, L. Tang, X. Liu, Z. Yao, S. Yi, J. Shao, J. Yan, S. Wang, H. Li, and X. Wang, "Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification," in *ICCV*, 2017.

[76] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *ICCV*, 2019.

[77] Y. Zhou and L. Shao, "Aware attentive multi-view inference for vehicle re-identification," in *CVPR*, 2018.

[78] X. Sun, Y. Hou, W. Deng, H. Li, and L. Zheng, "Ranking models in unlabeled new environments," in *ICCV*, 2021, pp. 11 761–11 771.

[79] Y. Wang, S. Liao, and L. Shao, "Surpassing real-world source training data: Random 3d characters for generalizable person re-identification," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3422–3430.

[80] K. Yan, Y. Tian, Y. Wang, W. Zeng, and T. Huang, "Exploiting multi-grain ranking constraints for precisely searching visually-similar vehicles," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 562–570.

[81] Y. Lou, Y. Bai, J. Liu, S. Wang, and L. Duan, "Veri-wild: A large dataset and a new method for vehicle re-identification in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3235–3243.

[82] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," in *ICME*, 2016.

[83] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *CVPR*, 2019.

[84] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[85] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[86] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[87] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018.

[88] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[89] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011.

[90] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[91] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 152–159.

[92] A. Das, A. Chakraborty, and A. K. Roy-Chowdhury, "Consistent re-identification in a camera network," in *European conference on computer vision*. Springer, 2014, pp. 330–345.

[93] L. Ma, H. Liu, L. Hu, C. Wang, and Q. Sun, "Orientation driven bag of appearances for person re-identification," *arXiv preprint arXiv:1605.02464*, 2016.

[94] Y.-J. Cho and K.-J. Yoon, "Pamm: Pose-aware multi-shot matching for improving person re-identification," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3739–3752, 2018.

[95] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[96] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *ECCV*, 2016.

[97] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *ECCV*, 2018.

[98] S. He, H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang, "Transreid: Transformer-based object re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[99] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *CVPR Workshops*, 2019.

[100] L. Song, C. Wang, L. Zhang, B. Du, Q. Zhang, C. Huang, and X. Wang, "Unsupervised domain adaptive re-identification: Theory and practice," *Pattern Recognition*, vol. 102, p. 107173, 2020.

[101] Y. Ge, D. Chen, and H. Li, "Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification," in *Proceedings of the International Conference on Learning Representations*, 2020.

[102] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, pp. 1–20, 2018.

[103] N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," in *ICLR*, 2019.

[104] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *CVPR Workshops*, 2018.

[105] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *CVPR*, 2017.

[106] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[107] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional learning for domain adaptation of semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6936–6945.

[108] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang, "Crdoco: Pixel-level domain transfer with cross-domain consistency," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1791–1800.

[109] W. M. Kouw, L. J. Van Der Maaten, J. H. Krijthe, and M. Loog, "Feature-level domain adaptation," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5943–5974, 2016.

[110] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *ICCV*, 2019.

[111] Y. Zou, X. Yang, Z. Yu, V. Kumar, and J. Kautz, "Joint disentangling and adaptation for cross-domain person re-identification," in *European Conference on Computer Vision*, 2020.

[112] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[113] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[114] Z. Xue, W. Mao, and L. Zheng, "Learning to simulate complex scenes for street scene segmentation," *IEEE Transactions on Multimedia*, vol. 24, pp. 1253–1265, 2021.

[115] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[116] Y. Hou, L. Zheng, and S. Gould, "Multiview detection with feature perspective transformation," *arXiv preprint arXiv:2007.07247*, 2020.

[117] S. Xiang, Y. Fu, G. You, and T. Liu, "Unsupervised domain adaptation through synthesis for person re-identification," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2020, pp. 1–6.

[118] 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 baselines," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[119] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from synthetic data for crowd counting in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8198–8207.

[120] A.-D. Doan, A. M. Jawaid, T.-T. Do, and T.-J. Chin, "G2d: from gta to data," *arXiv preprint arXiv:1806.07381*, 2018.

[121] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[122] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford *et al.*, "Robothor: An open simulation-to-real embodied ai platform," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3164–3174.

[123] M. Mueller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, 08 2017.

[124] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[125] M. Mozifian, A. Zhang, J. Pineau, and D. Meger, "Intervention design for effective sim2real transfer," *arXiv preprint arXiv:2012.02055*, 2020.

[126] R. K. Jones, T. Barton, X. Xu, K. Wang, E. Jiang, P. Guerrero, N. J. Mitra, and D. Ritchie, "Shapeassembly: Learning to generate programs for 3d shape structure synthesis," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–20, 2020.

[127] K. Yin, J. Gao, M. Shugrina, S. Khamis, and S. Fidler, "3dstylenet: Creating 3d shapes with geometric and texture style variations," in *ICCV*, 2021, pp. 12 456–12 465.

[128] T. Shi, Y. Yuan, C. Fan, Z. Zou, Z. Shi, and Y. Liu, "Face-to-parameter translation for game character auto-creation," in *ICCV*, 2019, pp. 161–170.

[129] Y. Zhang, M. Hassan, H. Neumann, M. J. Black, and S. Tang, "Generating 3d people in scenes without people," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6194–6204.

[130] D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler, "Neural parts: Learning expressive 3d shape abstractions with invertible neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3204–3215.

[131] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[132] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV Workshops*, 2016.

[133] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[134] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Advances in Neural Information Processing Systems*, 2019.

[135] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[136] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camstyle: A novel data augmentation method for person re-identification," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1176–1190, 2018.

[137] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[138] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[139] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[140] N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," in *Proceedings of the International Conference on Learning Representations*, 2018.

[141] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[142] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[143] W. Deng, L. Zheng, Q. Ye, Y. Yang, and J. Jiao, "Similarity-preserving image-image domain adaptation for person re-identification," *arXiv preprint arXiv:1811.10551*, 2018.

[144] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camstyle: A novel data augmentation method for person re-identification," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1176–1190, 2019.

[145] Z. Zheng, T. Ruan, Y. Wei, and Y. Yang, "Vehiclenet: Learning robust feature representation for vehicle re-identification," in *CVPR Workshops*, 2019.

[146] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *ICCV*, 2017.

[147] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6028–6039.

[148] Z. Zhuang, L. Wei, L. Xie, T. Zhang, H. Zhang, H. Wu, H. Ai, and Q. Tian, "Rethinking the distribution gap of person re-identification with camera-based batch normalization," in *European Conference on Computer Vision*. Springer, 2020, pp. 140–157.

[149] X. Liu, S. Zhang, Q. Huang, and W. Gao, "Ram: a region-aware deep model for vehicle re-identification," in *ICME*, 2018.

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

[151] R. Chu, Y. Sun, Y. Li, Z. Liu, C. Zhang, and Y. Wei, "Vehicle re-identification with viewpoint-aware metric learning," in *ICCV*, 2019.

[152] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[153] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[154] W. Deng and L. Zheng, "Are labels always necessary for classifier accuracy evaluation?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 069–15 078.

[155] W. Deng, S. Gould, and L. Zheng, "What does rotation prediction tell us about classifier accuracy under varying testing environments?" in *International Conference on Machine Learning*, 2021.

[156] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[157] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[158] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[159] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain–computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

[160] M. A. Lebedev and M. A. Nicolelis, "Brain–machine interfaces: past, present and future," *TRENDS in Neurosciences*, vol. 29, no. 9, pp. 536–546, 2006.

[161] F. Su, L. Xia, A. Cai, Y. Wu, and J. Ma, "Eeg-based personal identification: from proof-of-concept to a practical system," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 3728–3731.

[162] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, and O. Kavehei, "A generalised seizure prediction with convolutional neural networks for intracranial and scalp electroencephalogram data analysis," *arXiv preprint arXiv:1707.01976*, 2017.

[163] F. Ebrahimi, M. Mikaeili, E. Estrada, and H. Nazeran, "Automatic sleep stage classification based on eeg signals by using neural networks and wavelet packet coefficients," in *Engineering in medicine and biology society, 2008. EMBS 2008. 30th annual international conference of the IEEE*. IEEE, 2008, pp. 1151–1154.

[164] S. Palazzo, C. Spampinato, I. Kavasidis, D. Giordano, and M. Shah, "Generative adversarial networks conditioned by brain signals," *PDF available on ucf. edu*, 2017.

[165] U. Orhan, K. E. Hild, D. E. II, B. Roark, B. Oken, and M. Fried-Oken, "Rsvp keyboard: An eeg based typing interface," in *Proceedings of the... IEEE International Conference on Acoustics, Speech, and Signal Processing/sponsored by the Institute of Electrical and Electronics Engineers Signal Processing Society. ICASSP*. NIH Public Access, 2012.

[166] S. Gandhi, T. Oates, T. Mohsenin, and D. Hairston, "Denoising time series data using asymmetric generative adversarial networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 285–296.

[167] J. A. Urigüen and B. Garcia-Zapirain, "Eeg artifact removal—state-of-the-art and guidelines," *Journal of neural engineering*, vol. 12, no. 3, p. 031001, 2015.

[168] S.-Y. Shao, K.-Q. Shen, C. J. Ong, E. P. Wilder-Smith, and X.-P. Li, "Automatic eeg artifact removal: a weighted support vector machine approach with error correction," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 2, pp. 336–344, 2009.

[169] Y. Liu, Y. Yao, W. Zhengjie, J. Plested, and T. Gedeon, "Generalized alignment for multimodal physiological signal learning," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[170] A. Schlögl, "An overview on data formats for biomedical signals," in *World Congress on Medical Physics and Biomedical Engineering, September 7-12, 2009, Munich, Germany*. Springer, 2009, pp. 1557–1560.

[171] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[172] V. Lawhern, A. Solon, N. Waytowich, S. Gordon, C. Hung, and B. Lance, "Eegnet: a compact convolutional neural network for eeg-based brain-computer interfaces," *Journal of neural engineering*, vol. 15, no. 5, pp. 056 013–056 013, 2018.

[173] M. Schröder, T. N. Lal, T. Hinterberger, M. Bogdan, N. J. Hill, N. Birbaumer, W. Rosenstiel, and B. Schölkopf, "Robust eeg channel selection across subjects for brain-computer interfaces," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 3103–3112, 2005.

[174] Z. Wang, R. M. Hope, Z. Wang, Q. Ji, and W. D. Gray, "Cross-subject workload classification with a hierarchical bayes model," *NeuroImage*, vol. 59, no. 1, pp. 64–69, 2012.

[175] K. Sundararajan, "Privacy and security issues in brain computer interfaces," Ph.D. dissertation, Auckland University of Technology, 2018.

[176] J. Kulynych, "Legal and ethical issues in neuroimaging research: human subjects protection, medical privacy, and the public communication of research results," *Brain and cognition*, vol. 50, no. 3, pp. 345–357, 2002.

[177] V. Robinson and E. B. Varghese, "A novel approach for ensuring the privacy of eeg signals using application-specific feature extraction and aes algorithm," in *Inventive Computation Technologies (ICICT), International Conference on*, vol. 2. IEEE, 2016, pp. 1–6.

[178] S. Al-Janabi, I. Al-Shourbaji, M. Shojafar, and S. Shamshirband, "Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications," *Egyptian Informatics Journal*, vol. 18, no. 2, pp. 113–122, 2017.

[179] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017.

[180] W. Liu, W.-L. Zheng, and B.-L. Lu, "Emotion recognition using multimodal deep learning," in *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part II 23*. Springer, 2016, pp. 521–529.

[181] H. Yang, S. Sakhavi, K. K. Ang, and C. Guan, "On the use of convolutional neural networks and augmented csp features for multi-class motor imagery of eeg signals classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 2620–2623.

[182] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *International conference on web-age information management*. Springer, 2014, pp. 298–310.

[183] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, "Learning representations from eeg with deep recurrent-convolutional neural networks," *arXiv preprint arXiv:1511.06448*, 2015.

[184] W.-L. Zheng and B.-L. Lu, "Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks," *IEEE Transactions on autonomous mental development*, vol. 7, no. 3, pp. 162–175, 2015.

[185] Y. Li, K. Dzirasa, L. Carin, D. E. Carlson *et al.*, "Targeting eeg/lfp synchrony with neural nets," in *Advances in Neural Information Processing Systems*, 2017, pp. 4620–4630.

[186] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of eeg motor imagery signals," *Journal of neural engineering*, vol. 14, no. 1, p. 016003, 2016.

[187] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving face recognition utilizing differential privacy," *Computers & Security*, vol. 97, p. 101951, 2020.

[188] H. Kiya, T. Nagamori, S. Imaizumi, and S. Shiota, "Privacy-preserving semantic segmentation using vision transformer," *Journal of Imaging*, vol. 8, no. 9, p. 233, 2022.

[189] T. Gedeon, J. Catalan, and J. Jin, "Image compression using shared weights and bidirectional networks," in *Proc. 2nd International ICSC Symposium on Soft Computing (SOCO'97)*, 1997, pp. 374–381.

[190] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[191] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, "Deep feature learning for eeg recordings," *arXiv preprint arXiv:1511.04306*, 2015.

[192] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[193] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[194] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[195] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[196] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[197] P. Sykacek and S. J. Roberts, "Adaptive classification by variational kalman filtering," in *Advances in Neural Information Processing Systems*, 2003, pp. 753–760.

[198] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in bioinformatics*, vol. 18, no. 5, pp. 851–869, 2017.

[199] P. A. Abhang and B. W. Gawali, "Correlation of eeg images and speech signals for emotion analysis," *British Journal of Applied Science & Technology*, vol. 10, no. 5, pp. 1–13, 2015.

[200] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[201] D. O'Reilly, M. A. Navakatikyan, M. Filip, D. Greene, and L. J. Van Marter, "Peak-to-peak amplitude in neonatal brain monitoring of premature infants," *Clinical neurophysiology*, vol. 123, no. 11, pp. 2139–2153, 2012.

[202] W.-L. Zheng, J.-Y. Zhu, Y. Peng, and B.-L. Lu, "Eeg-based emotion classification using deep belief networks," in *2014 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2014, pp. 1–6.

[203] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[204] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[205] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," in *Advances in neural information processing systems*, 2018, pp. 700–709.

[206] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[207] T. Ge, F. Wei, and M. Zhou, "Reaching human-level performance in automatic grammatical error correction: An empirical study," *arXiv preprint arXiv:1807.01270*, 2018.

[208] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[209] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[210] Z. Liu, H. Jin, T.-H. Wang, K. Zhou, and X. Hu, "Divaug: plug-in automated data augmentation with explicit diversity maximization," in *ICCV*, 2021, pp. 4762–4770.

[211] Y. Yao, X. Tian, Z. Tang, S. Biswas, H. Lei, T. Gedeon, and L. Zheng, "Training with product digital twins for autoretail checkout," *arXiv preprint arXiv:2308.09708*, 2023.

[212] L. Yang, P. Luo, C. Change Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *CVPR*, 2015.