UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Generative Models with Human Preferences

by Yinghua Yao

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Ivor W. Tsang and Prof. Xin Yao

University of Technology Sydney
Faculty of Engineering and Information Technology

May 2023

Certificate of Original Authorship

I, Yinghua Yao, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with the Southern University of Science and Technology.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 8 May 2023

Dedicated to my family

Acknowledgements

First and foremost, I am tremendously grateful to my supervisors Prof. Xin Yao and Prof. Ivor Tsang. They have consistently provided guidance with their profound knowledge and extensive research experience throughout my PhD study. Prof. Yao is the one who led me into the realm of scientific research, beginning with his mentorship on my undergraduate graduation project, and then affording me an opportunity to continue my research journey. His research philosophy "a good research question is half of the research work" cultivated my taste in research. I have yet to meet someone whose mind is as brimming with research ideas as Prof. Tsang's. I am so impressed by his exceptional understanding of problems and his talent for discovering unexpected links between research works. What I learned from him has fueled my passion for research further.

My sincere gratitude also goes to Dr. Yuangang Pan, who helped me to realize my potential for research. Thanks to his assistance and companionship, I was able to navigate my adventure more smoothly. I consider myself incredibly fortunate to have commenced my PhD journey simultaneously with my friend Jing Li. We encourage and collaborate with each other throughout the journey. I would like to thank my female friends, Dr. Yan Zhang, Dr. Lu Zhang, Shuyi Zhang, Xinming Shi, Kim, and Sel, whom I met on this journey. The close relationship with them has brought a lot of joy and encouragement to my research and life. I thank Dr. Yaxin Shi, Dr. Xiaowei Zhou, and Xingrui Yu for weekly discussions and daily meal chats during my first PhD year. I thank Dashan Gao and Yunce Zhao for their collaboration on the research project. I also appreciate people met at University of Technology Sydney (UTS), Dr. Zhibin Li, Prof. Ling Chen, Prof. Yarui Chen, Prof. Weijie Chen, Dr. Xu Chen, Dr. Yueming Lyu, Jinliang Deng, Peiyao Zhao, Feiyang Ye, Bowen Xing, Yujie Fang, Cheng Chen, Ken, Zijian Lei and Lefei Zhang. Thanks also to people from Southern University of Science and Technology (SUSTech), Dr. Weijie Zheng, Dr. Liyan Song, Dr. Changwu Huang, Dr. Zilu Wang, Yuannan Ji, Weiying Pan, Huimin Zheng, Shuxian Li, Gan Ruan, Qinya Li, and Zhi Cao for my joyful time in 319 lab.

Many thanks to UTS and SUSTech for providing a conducive research environment and to A*STAR CFAR for offering me a valuable internship.

Last but not least, I owe the deepest debt of gratitude to my sister and my parents for all the years of their love and support.

Abstract

Powered by the learning capacity of deep neural networks, generative models have facilitated the scalable modeling of complex, high-dimensional data and are extensively used in various fields. In practical scenarios, deep generative models (DGMs) are often required not only to produce authentic samples but also to optimize synthetic samples for some desired properties. While existing DGMs are capable of generating data meeting users' expectations using desired class/attribute labels or an off-shelf evaluator, acquiring complete knowledge pertaining to the target property is an indispensable prerequisite for obtaining the labels or the evaluator, which is unmet in many real-world applications. In addition, discrete labels have limited description capacity, which cannot capture intra-category differences.

This thesis resorts to human preferences that are more readily accessible, which are typically represented by comparisons among a list of samples and can provide fine-grained information. Motivated by real-world problems, preferences-guided desired data generation can be defined in terms of the dataset level or the instance level, which means generating the desired data based on a given dataset or a single sample, respectively. Particularly, a preference-guided generative model at the dataset level can automate the design of biological data or generate images that align with user interests; a preference-guided generative model at the instance level can facilitate tasks like style transfer and facial expression generation. This thesis focuses on deep generative modeling from human preferences in different scenarios. Specifically,

- *First investigation on preference-guided desired data generation at the dataset level*. We incorporate pairwise preferences into the existing framework of DGMs to generate high-quality desired data, i.e., part of the training dataset. To be specific, we introduce an additional pairwise ranking loss over the critic of Wasserstein Generative Adversarial Network, which slightly shifts the learned distribution of the generative model towards the desired data distribution. Our model converges to the desired data distributions by multi-step distribution shifts.

- *A new and more efficient generative modeling paradigm for preference-guided desired data generation at the dataset level.* We introduce a new generative modeling paradigm

to learn the desired data distribution from partial preferences. Specifically, we design an adversarial ranking framework, which is proven to estimate a relativistic f-divergence between the desired data distribution and the generated data distribution. This approach shifts the generative model's distribution towards the desired data distribution in a single step, resulting in reduced training expenses.

- *Preference-guided desired data generation at the instance level.* We propose an adversarial ranking paradigm for generating desired data for single input samples based on comparisons in terms of specified attributes. In particular, we aim to generate a series of realistic versions of the input image with smooth changes on the attributes, a.k.a., high-quality fine-grained image-to-image (I2I) translation. The adversarial training between the ranker and the generator enhances the ability of the ranker and encourages a better generator. Meanwhile, our ranker enforces a linearizedly continuous change between the generated image and the input image, which promotes a better fine-grained control over the interested attribute.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This thesis focuses on deep generative modeling from human preferences. Preference is a natural expression of user opinions [Fürnkranz and Hüllermeier, 2017], which can be used to align generative models with human needs [Yao et al., 2022, Yu and Kovashka, 2020]. This chapter briefly introduces the background, the research problems studied in this thesis, and the contributions of this thesis.

1.1 Background

Intelligent agents are constantly generating, acquiring, and processing data. Generative models, one of the subfields of artificial intelligence, view the world under the lens of probability. In such a worldview, we can think of any kind of observed data, called $\mathcal{X}$, as a finite set of samples from an underlying distribution, called P_r. At its very core, the goal of generative models is to approximate this data distribution with a chosen model P_{g} given access to the dataset $\mathcal{X}$. The learned model P_{g} can be used for data sampling or data transformation [Goodfellow et al., 2014, Zhu et al., 2017]. Recent advances in parameterizing these models using deep neural networks [Goodfellow et al., 2016], combined with progress in stochastic optimization methods [Kingma and Ba, 2015], have enabled scalable modeling of complex, high-dimensional data, such as images [Zhu et al., 2017], video [Zhou et al., 2021], speech [Yi and Mak, 2020] and text [Yang et al., 2020].

Despite the great success of deep generative models, these models themselves have inherent limitations. For examples, generative adversarial networks (GANs) suffer from the mode collapse issue [Che et al., 2017]. Variational autoencoders (VAEs) are deficient for the poor data quality [Bredell et al., 2023]. More importantly, while vanilla generative modeling is an interesting technical problem, it has limited practical relevance, especially for those generative models that are only capable of generating data [Goodfellow et al., 2014] rather

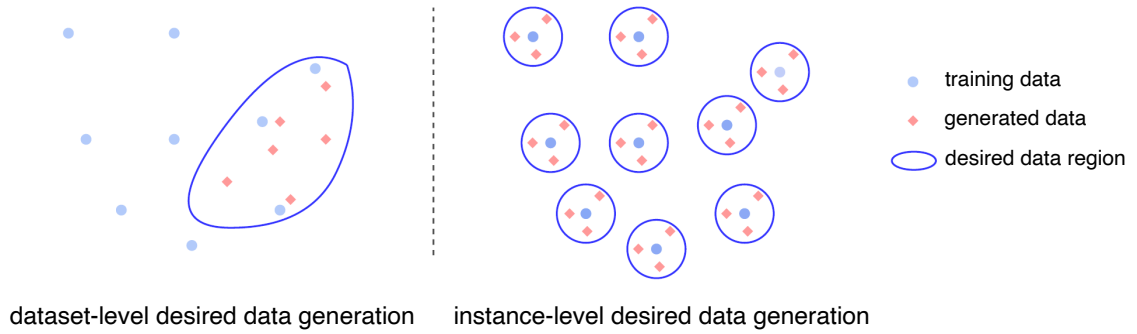


Figure 1.1 Desired data generation in terms of the dataset level and the instance level. The former scenario aims to learn the distribution of user-desired data when only partial instead of the entire dataset possesses the desired properties, while the latter one targets at generating a series of realistic versions of each sample with desired changes on some specific attributes.

than providing an estimate of the density function. After all, such models merely provide more data (e.g., images), and the world has no shortage of data observations under the prosperity of the Internet. In contrast, controllable generative models can generate data to match some user-desired properties, meeting the needs of many real-world applications [Engel et al., 2018]. Fig. 1.1 displays two scenarios of desired data generation. DGMs for inverse material design [Sanchez-Lengeling and Aspuru-Guzik, 2018] is a typical example of desired data generation in terms of the dataset level, where the DGMs model the material space from a database of existing materials and the generative process would be controlled towards desirable properties (See Section 4.7.3 for case studies). Desired data generation in terms of the instance level can be instantiated as the fine-grained image-to-image (I2I) translation task [Wu et al., 2019b], which translates an input image into the desired ones with changes in some specified attributes (See Section 6.4).

Great success has been driven in the field of using some predefined attribute/class labels or a well-specified evaluator to direct the generation process of deep generative models (DGMs) [Mirza and Osindero, 2014, Asokan and Seelamantula, 2020, Engel et al., 2018, Gupta and Zou, 2019, Samanta et al., 2020]. However, in many real-world applications, human may not be able to explicitly articulate the data they are interested in via the desired class/attribute labels or high rewards from the evaluator. Instead, more naturally, human would often implicitly describe their preferences for data by ranking two or more samples in terms of certain interested properties [Fürnkranz and Hüllermeier, 2017]. More importantly, labels/evaluators are argued to be more restricted than preferences. To be specific,

- *Complete vs. partial knowledge about the properties.* Deriving the desired data via labels/evaluators requires complete knowledge about the target properties, which is expensive and is not available in many real-world applications [Christiano et al., 2017].

In contrast, preferences can be easily collected with partial ranking relation in terms of the interested properties among small subsets of dataset. For example, when asked to produce images of young faces, DGMs can model a conditional distribution conditioned on "young" labels or be optimized to only generate images for which an age evaluator output small values. Nevertheless, either the labeling of "young" faces or the age evaluator relies on facial feature information across all ages. Instead, preferences simply involve comparisons of facial features among several images. Therefore, it is not necessary to label all training data when resorting to preferences, but not when relying on labels/evaluators.

- *Coarse vs. fine-grained.* Class labels or attribute labels have limited description capacity [Parikh and Grauman, 2011] since such category-level labels cannot capture intra-category differences. Instead, human preferences can be used to describe fine-grained information [Parikh and Grauman, 2011, Chen et al., 2013, Raman and Joachims, 2014]. For example, people may feel tangled when asked to determine whether a face in an image is smiling or not. This is because there are many images that are difficult to categorize, aside from a small number of obvious smiling and non-smiling images. Instead, people can compare a set of images and rank them in terms of the strength of the smile attribute.

- *When desired data is insufficient.* Since generation conditioned on class labels is dominated by those classes with sufficient training observations, a class with limited samples would be overlooked by the generative model. But differently, generation guided by preferences can stress the modeling for "preferred class" (See Section 5.6.4 for empirical support).

Some works on instance-level desired data generation started to use human preferences to guide the generation of DGMs [Yu and Kovashka, 2020, Saquil et al., 2018]. However, they suffer from the conflict between the data quality of generation and the modeling for preferences, thus unable to achieve high-quality desired data generation. Therefore, in this thesis, we aim to develop new frameworks that can incorporate human preferences into DGMs while reconciling generation quality and the modeling for preferences. Particularly, we are the first one to apply human preferences to dataset-level desired data generation.

1.2 Research Questions

In this thesis, we focus on DGMs with human preferences to handle two scenarios of desired data generation (Fig. 1.1). We formulate concrete research questions as follows.

Research question 1: *How can we incorporate human preferences into the existing framework of DGMs in order to bias the generative process towards desired properties?*

We propose including preferences in the learning of Generative Adversarial Network GAN) [Goodfellow et al., 2014] due to its flexible formulation and advanced capacity of high-quality data generation. To be specific, Relativistic GAN (RGAN) [Jolicoeur-Martineau, 2019], a GAN variant that learns the distribution of given training data via an adversarial training between a critic and a generator, regards the critic values as the indicators of sample quality, which are analogous to ranking scores. Motivated by this, we consider taking the critic values as ranking scores that can reflect user preferences over samples, which is achieved by a ranking loss for pairwise preferences based on the critic values. Consequently, our new critic can guide the generator towards the desired data distribution. The proposed framework is called Differential-Critic GAN (DiCGAN).

As will be explained in Chapter 4, DiCGAN learns the desired data distribution by a sequence of minor corrections, where each correction shifts the distribution of generative model towards the desired data distribution slightly. With multi-step shifts, the new training samples (i.e., generated samples introduced into the training dataset after one minor correction) need to be manually annotated, increasing training costs. Therefore, we consider proposing a novel generative model that can align the distribution between the desired data and the generated data once for all as follows.

Research question 2: *Can we design a new preference guided generative modeling paradigm to learn a desired data distribution at the dataset level in single step, thereby avoiding annotation costs for generated samples?*

Motivated by the adversarial classification defined in GANs [Goodfellow et al., 2014] and learning to rank [Cao et al., 2007b], we propose an adversarial ranking game between a ranker and a generator, each with their own objectives about rankings based on preferences that favor generated samples differently. It turns out that the ranker estimates a distribution divergence between the desired data and the generated data, and the generator is to minimize the divergence to obtain the desired data distribution.

Beyond considering desired data generation at the dataset level, it naturally comes to the following research question.

Research question 3: *How can we achieve desired data generation at the instance level by using human preferences?*

In particular, we aim at generating a series of realistic versions of the input image only with desired changes on a certain specific attribute, a.k.a. fine-grained image-to-

image (I2I) translation. We propose another new adversarial ranking framework consisting of a ranker and a generator for this problem. The adversarial ranking process enhances the ability of the ranker and encourages a better generator. As a result, the generator can translate an input image to the desired counterpart with smooth subtle changes in terms of the interested attributes.

1.3 Thesis Contributions

In the following, we summarize the key contributions of this thesis.

1. We propose Differential-Critic GAN (DiCGAN) to learn the desired data distribution when only partial instead of the entire dataset possesses the desired properties. This is the first work that applies partial knowledge about the properties, i.e., human preferences, to desired data generation at the dataset level, which is superior over previous works that rely on labels/evaluators requiring comprehensive knowledge about the properties. DiCGAN converges to the distribution of desired samples by multi-step distribution shifts, where each step slightly biases the generative model towards the desired data distribution guided by the preferences. We provide theoretical and empirical support for the convergence. We apply DiCGAN to generate images that meet the user's interest on two authentic datasets and design biological products with desired properties on a gene dataset sourced from the real world. Our DiCGAN achieves better performance in learning the user-desired data distribution than various baselines, especially in the cases of insufficient desired data and limited supervision.

2. We propose a new generative model, called Generative Adversarial Ranking Net (GARNet), which can directly enable a distribution alignment between the generated data and the desired data under the context of human preferences. Thus, GARNet is a more efficient generative modeling paradigm than DiCGAN for preference-guided desired data generation at the dataset level, which can prevent additional annotation costs for synthetic samples through a single-step distribution shift. In particular, we design GARNet as an adversarial ranking game between a ranker and a generator, which turns out minimizing a divergence between the generated data distribution and the desired data distribution. We also provide empirical and theoretical support that the distribution learned by GARNet is determined by the ranking score vector given by the users, which can enable a more flexible user control. We apply GARNet to generate user-desired images in terms of variaous interested properties, i.e., discrete/continuous

property, single/multiple properties and to boost imbalance class learning by promoting the generation of minorities.

3. We propose Translation via RIval Preferences (TRIP) for fine-grained image-to-image translation task (namely, desired data generation at the instance level), which is the first framework that reconciles the goal for fine-grained translation and the goal for high-quality generation. TRIP also consists of an adversarial game between a ranker and a generator, which improves the capacity of the ranker and incentivizes the generator to produce high-quality outputs with smooth changes on the specified attributes. In addition, the ranker enforces a linearizedly continuous change between the generated image and the input image, which provides a more fine-grained control over the interested attributes. TRIP is applied to fine-grained facial expression generation and style transfer on real-world datasets, achieving the state-of-art results.

1.4 Thesis Outline

Prior to delving into the thesis topic on generative modeling from human preferences, we conduct some preliminary studies to investigate fundamental issues and applications of (controllable) DGMs. One study proposes a novel regularization to escape from mode collapse in GANs. Another proposes a conditional Variational Auto-Encoder (VAE) with disentanglement regularization that achieves better clustering results compared to vanilla DGMs. The two preliminary studies paved a basis for the exploration on generative modeling from human preferences. (1) In the first study, we will clarify that WGAN would not suffer from the mode collapse issue, so our DiCGAN built on it would maintain good sample diversity. (2) The second study preliminarily explores the potential of controllable DGMs compared to vanilla DGMs, highlighting the greater practical significance for our subsequent investigations.

About generative modeling from human preferences, in terms of the dataset level, we explore how to incorporate preferences into the existing framework of DGMs. We further develop a new and more efficient generative modeling paradigm to learn a desired data distribution from preferences. In terms of the instance level, we propose a new framework that achieves the state-of-art results on high-quality fine-grained image-to-image (I2I) translation. The thesis is organized as follows:

- Chapter 2 introduces the related work.
- Chapter 3 introduces some preliminary studies, namely, DGMs with diversity and disentanglement regularization.

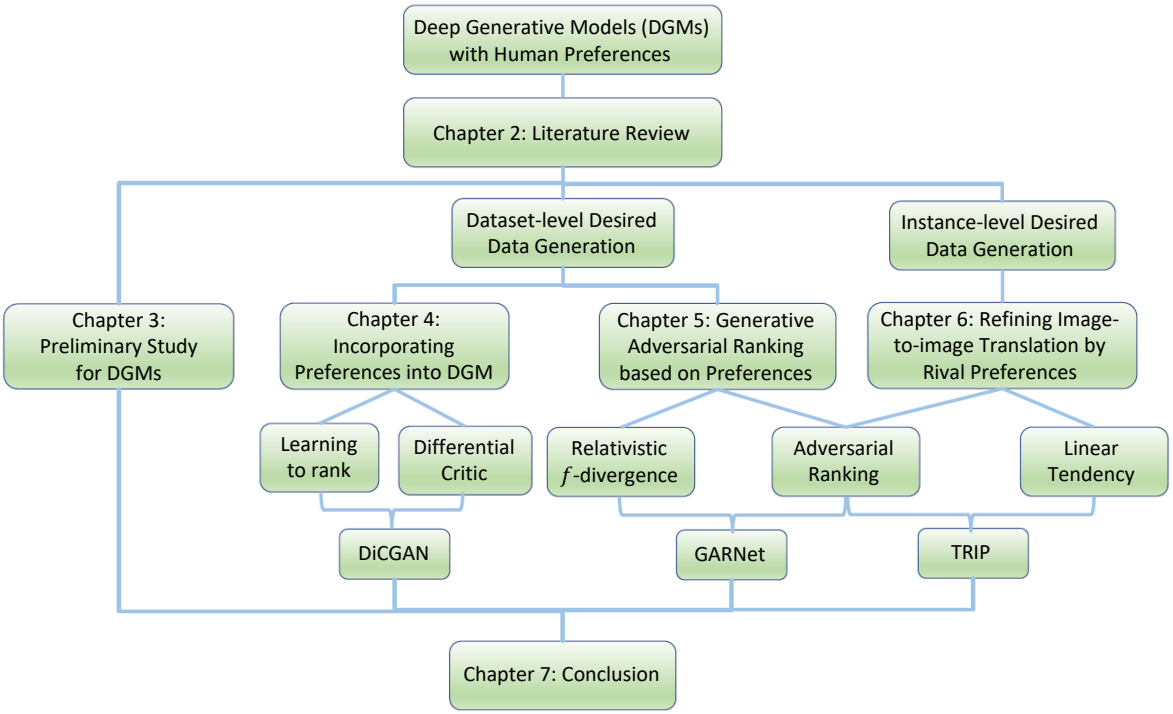


Figure 1.2 The organization of this thesis.

- Chapter 4 incorporates preferences into the existing framework of DGMs for dataset-level desired data generation, which relates to Contribution 1 in Section 1.3.
- Chapter 5 designs a new generative modeling paradigm that is more efficient for preference-guided desired data generation at the dataset level, which relates to Contribution 2 in Section 1.3.
- Chapter 6 develops an effective framework for instance-level desired data generation, especially for fine-grained image-to-image translation tasks, which relates to Contribution 3 in Section 1.3.
- Chapter 7 concludes the thesis.

The organization of this thesis is in Fig. 1.2.

1.5 Publications

The work present in this thesis resulted in a total of five papers (i.e., 1 to 5 listed in the following). Other listed works are collaborative works with peers during my PhD.

Works in Thesis

1. **Yinghua Yao**, Yuangang Pan, Jing Li, Ivor W. Tsang and Xin Yao. Generative Adversarial Ranking Nets. *Journal of Machine Learning Research*, 2023. Under Review.

2. **Yinghua Yao**, Yuangang Pan, Jing Li, Ivor W. Tsang, and Xin Yao. COUF: Clustering withOut the Unwanted Factor. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. Under Review.

3. **Yinghua Yao**, Yuangang Pan, Ivor W. Tsang, and Xin Yao. TRIP: Refining Image-to-Image Translation via Rival Preferences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. Major Revision.

4. **Yinghua Yao**, Yuangang Pan, Ivor W. Tsang, and Xin Yao. Differential-Critic GAN: Generating What You Want by a Cue of Preferences. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. doi: 10.1109/TNNLS.2022.3197313. The paper is available for early access at https://ieeexplore.ieee.org/document/9868048.

5. **Yinghua Yao**, Yuangang Pan, Ivor W. Tsang, and Xin Yao. Support Matching: A Novel Regularization to Escape from Mode Collapse in GANs. *International Conference on Neural Information Processing*, pages 40–48, Springer, 2019.

Table 1.1 Authors' contributions in my thesis works. Yinghua Yao (YY), Yuangang Pan (YP), Jing Li (JL), Ivor W. Tsang (IWT), Xin Yao (XY).

Paper	1	2	3	4	5
Conceptualization of the idea	YY	YY	YY	YY, YP, IWT	YY, IWT
Methodology	YY, YP	YY, YP, JL, XY	YY, YP, IWT	YY, YP	YY
Experimental study	YY				
Writing - original draft	YY				
Writing - comments/edits	all authors				
Supervision	IWT, XY				

Other Works

6. Jing Li, Yuangang Pan, Yueming Lyu, **Yinghua Yao**, Yulei Sui, and Ivor W. Tsang. Earning Extra Performance from Restrictive Feedbacks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. doi:10.1109/TPAMI.2023.3273249.

7. Yunce Zhao, Dashan Gao, **Yinghua Yao**, Zeqi Zhang, Bifei Mao and Xin Yao. Robust Deep Learning Models Against Semantic-Preserving Adversarial Attack. *International Joint Conference on Neural Networks*, 2023.

Chapter 2

Literature Review

This chapter discusses works related to this thesis. Specifically, Section 2.1 reviews two deep generative modeling frameworks that are studied in this thesis. Section 2.2 presents current relevant works on desired data generation and discusses research gaps. Section 2.3 introduces techniques to learn from human preferences.

2.1 Deep Generative Models (DGMs)

Generative models learn to capture the statistical distribution of training data, allowing to synthesize samples from the learned distribution. Traditional methods, such as SMOTE [Chawla et al., 2002], MWMOTE [Barua et al., 2012] for replicating data distribution through synthetic oversampling, Gaussian mixture models [Bishop, 2006] for maximizing data likelihood, score matching [Hyvärinen et al., 2009] and noise-contrastive estimation [Gutmann and Hyvärinen, 2010] for specifying learned probability density analytically up to a normalization constant, as well as Bayesian networks (belief networks) [Heckerman, 2008] representing variables and their conditional dependencies through a directed acyclic graph, have been studied over the past decades. In recent years, equipped with deep neural networks, generative models, e.g., Generative Adversarial Networks (GANs) [Goodfellow et al., 2014], Variational Autoencoders (VAEs) [Kingma and Welling, 2014], normalizing flows [Papamakarios et al., 2021], diffusion models [Ho et al., 2020], have achieved tremendous success in modeling complex, high-dimensional natural signals. In particular, this thesis involves GANs and VAEs, which are two of the most commonly used and efficient deep generative models. We introduce basic knowledge about them in the following.

GAN [Goodfellow et al., 2014] performs generative modeling by learning a generator G that maps low-dimensional latent space $\mathcal{Z}$ to data space $\mathcal{X}$, i.e., $G : \mathcal{Z} \to \mathcal{X}$, given samples from the training data distribution, namely, $x \sim P_{\mathrm{r}}(x)$. The goal is to find G that achieves

Table 2.1 Main mathematical notations in this thesis.

Notation	Explanation
$\mathcal{X}$	data space/training data $\mathcal{X} = \{x_n\}_{n=1}^N$
$\mathcal{Z}$	latent space
$\mathcal{Y}$	class labels $\mathcal{Y} = \{y_1, y_2, \ldots, y_T\}$ endowed with an order $y_1 > y_2 > \ldots > y_T$
$\mathcal{O}$	score space for a continuous property
$y(x)$	function that maps sample x to its class label
$o(x)$	function that maps sample x to its underlying ground-truth ranking score
$P_{\mathrm{r}}(x)$	the distribution of training data
$P_{\mathrm{d}}(x)$	the distribution of user-desired data
$P_{\mathrm{g}}(x)$	the distribution of generated data
$P(z)$	the distribution of input noise
$x_1 \succ x_2$	x_1 is preferred over x_2, $x_1, x_2 \in \mathcal{X}$
s	preferences, $s := s^1 > s^2 > \ldots > s^l, s^i \in \mathcal{X}$
x_{g}	generated sample, $x_{\mathrm{g}} \sim P_{\mathrm{g}}$
$\mathcal{S}$	a collection of pairwise/list preferences over training data $\mathcal{X}$
$\pi(s)$	ground-truth score vector for preference s
$s^{(R)}$	target preference to train the ranker, i.e., $s^{(R)} := s^1 > \ldots > s^l > x_{\mathrm{g}}$
$s^{(G)}$	target preference to train the generator, i.e., $s^{(G)} := x_{\mathrm{g}} > s^1 > \ldots > s^l$
v	continuous attribute variable
G	generator
D	discriminator/critic
R	ranker

$P_{\mathrm{g}}(x) = P_{\mathrm{r}}(x)$, where $P_{\mathrm{g}}(x)$ is the distribution of fake data $x = G(z)$. In order to train the generator G, GAN introduces another network, i.e., discriminator D, to discriminate real data from fake data. The generator G is trained to produce images that are conceived to be realistic by the discriminator D. Two networks are trained alternately until the generator successfully fools the discriminator. Vanilla GAN's objective is defined as follows:

$$\min_G \max_D \mathbb{E}_{P_{\mathrm{r}}(x)} \left[\log \sigma\left(D(x)\right)\right] + \mathbb{E}_{P_{\mathrm{g}}(x)} \left[\log\left(1 - \sigma\left(D(x)\right)\right)\right], \tag{2.1}$$

where $\sigma\left(D(x)\right)$ is the probability that the input data is real and σ is the sigmoid function.

WGAN [Arjovsky et al., 2017b] is a stable variant of GANs defining the loss functions in terms of the non-transformed discriminator D, called critic. Specifically, WGAN measures the quality of fake data in terms of the Wasserstein distance (W-distance) between the real data distribution and the fake data distribution. The W-distance is approximated by the difference in the average critic values between the real data and the fake data. WGAN's

objective is defined as follows:

$$\min_G \max_D \mathbb{E}_{P_r(x)}[D(x)] - \mathbb{E}_{P_g(x)}[D(x)], \tag{2.2}$$

where D is the critic enforced with a 1-Lipschitz constraint [Gulrajani et al., 2017]. *Note our work in Chapter 4 is developed based on WGAN by building the ranking on its critic.*

While owning distinguished abilities in generating sharp images, GANs suffer from several challenging problems. Mode collapse is one of the major challenges which refers to poor diversity in generated samples [Hong et al., 2019].

Work 1: In our preliminary study (Section 3.1), we target at solving this issue and propose a new GAN variant to promote mode diversity.

VAE [Kingma and Welling, 2014] learns a generative model that can be used to generate new samples by sampling from the learned latent space distribution. It uses a probabilistic approach that involves learning the parameters of two probability distributions: the encoder's and the decoder's output distribution. The encoder maps the input data to a distribution over the latent space, and the decoder maps the latent space back to the data space. Its training objective is to maximize the variational lower bound (VLBO) of the data log-likelihood, i.e., the right side of the following equation:

$$\mathbb{E}_x[\log P(x)] \geq \mathbb{E}_x\left[\mathbb{E}_{z\sim Q(z|x)}[\log P(x \mid z)] - KL[Q(z \mid x)\|P(z)]\right]. \tag{2.3}$$

$P(x)$ is short for the training data distribution $P_r(x)$ here. In the VLBO, the first term is a reconstruction loss that measures the difference between the input data and its reconstruction; the second term is a regularization loss that encourages the latent space distribution to be close to a prior distribution $P(z)$.

2.2 Controllable DGMs for Desired Data Generation

While vanilla generative models lack the ability to control the generation of data and thus have limited practical significance, various controllable generative models fill in the gaps and realize greater potential in real-world applications [Engel et al., 2018].

Work 2: Our preliminary study (Section 3.2) explores that controllable VAEs can achieve better clustering results compared to vanilla DGMs.

Controllable DGMs can generate data that matches specific properties, also referred to simply as "desired data". Properties can be either discrete or continuous. For a discrete property, desired data usually refers to some specific classes of data. For a continuous property, desired data can be those with high evaluation values w.r.t. the property. As

discussed in Fig. 1.1, controllable DGMs for desired data generation can be considered either at the dataset level as well as at the instance level. We particularly review related work in terms of these two aspects in the following, so as to motivate our main work of this thesis.

2.2.1 Dataset-level Desired Data Generation

Controllable DGMs based on Labels/Functions

Vanilla DGMs, like GANs [Goodfellow et al., 2014, Arjovsky et al., 2017b], VAEs [Kingma and Welling, 2014], can be adapted to learn a user-desired data distribution. A naive way is to first select the samples possessing the desired property, which can be desired classes of data or data with high evaluation values w.r.t. the properties. Then we perform regular GAN training only on the selected samples to derive the desired data distribution. However, constructing a desired subset as training data relies on complete knowledge about the properties, which may not be accessible in real applications (explained in Section 1.1 in details). Even, such methods would fail when there are insufficient desired samples.

Conditional DGMs [Mirza and Osindero, 2014, Odena et al., 2017, Miyato and Koyama, 2018, Sohn et al., 2015, Li et al., 2020b, Pumarola et al., 2020, Ho and Salimans, 2022] derived desired data distribution via a conditional distribution conditioned on desired class/attribute labels. However, the labeling of desired data and undesired data also requires complete knowledge about the properties. On the other hand, the generation performance of condition-based GAN is governed by the respective conditions with sufficient training observations. When the desired data is limited, the conditional modeling is dominated by the major classes, i.e., undesired data, resulting in a failure to capture the desired data distribution.

A few works [Engel et al., 2018, Gupta and Zou, 2019, Samanta et al., 2020] proposed to apply a ready-to-use evaluator for the target property to control the generation of generative models, where the synthetic data are optimized to have high evaluation values. However, the evaluator may not exist in real-world applications [Christiano et al., 2017] since it necessitates complete knowledge about the property.

Work 3&4: Chapter 4 is the first work targeting dataset-level desired data generation by using human preferences that are more accessible and require less supervision since it is simply partial knowledge about the property (see Section 1.1 for details). Further, we propose a new generative modeling paradigm, which is more efficient for preference-guided desired data generation in Chapter 5.

2.2.2 Instance-level Desired Data Generation

Since this thesis particularly focuses on the fine-grained image-to-image (I2I) translation task for desired data generation at the instance level, we review related work on this task in the following.

Controllable DGMs based on Binary Attributes

Many works conducted fine-grained I2I translation based on coarse binary attributes that indicate the presence (or absence) of some certain attributes in an image. Though it is feasible to interpolate the binary attributes, the interpolation quality is unsatisfactory since the models are only trained on binary-valued attributes and thus the interpolation is ill-defined [Berthelot et al., 2018, Wu et al., 2019b]. Apart from the defect of using binary attributes, the following works have other shortcomings inherited from the adopted generative models.

Autoencoder-based methods can provide a good latent representation of the input image. Some works [Lample et al., 2017, Liu et al., 2018, Li et al., 2020b, Ding et al., 2020] proposed to disentangle the attribute-dependent latent variable from the image representation based on binary attributes but resorted to different disentanglement strategies. Then the fine-grained translation can be derived by smoothly manipulating the attribute variable of the input image. However, the reconstruction loss, which is used to ensure the image quality, cannot guarantee a high fidelity of the interpolated images.

Flow-based works proposed incorporating feature disentanglement mechanism into flow-based generative models [Kondo et al., 2019]. However, the designed multi-scale disentanglement requires large computation. Liu et al. [2019] applied an encoder to map the attribute space to the latent space of flow-based model by resorting to a binary attribute classifier, which however fails to capture fine-grained attribution information.

GAN-based methods are widely adopted frameworks for high-quality image generation. He et al. [2019], Lin et al. [2021] directly modeled the attributes with binary classification, which cannot capture fine-grained attribute information, and hence fail to make a smooth control over the attributes. Wu and Lu [2020] maximized mutual information between the interpolated attribute code and the generated images for fine-grained facial expression, but the experimental results show the loss of facial details and identity information in the synthesized expression.

Diffusion model based methods were mainly focusing on binary I2I translation [Li et al., 2023, Kwon and Ye, 2023, Zhao et al., 2022, Saharia et al., 2022]. To our best knowledge, there is not published work on fine-grained I2I translation at the time of writing this thesis.

Controllable DGMs based on Other Priors

Deng et al. [2020] embedded 3D priors into adversarial learning. However, it relies on available priors for attributes, which limits the practicality. Alharbi and Wonka [2020] proposed an unsupervised disentanglement method. It injects the structure noises to GAN for controlling specific parts of the generated images, which makes global or local features changed in a disentangled way. However, it is unclear how global or local features are related to facial attributes. Thus, it is difficult to change specific attributes. Some studies [Pumarola et al., 2018, Ling et al., 2020, Xia et al., 2021] achieved fine-grained facial expression generation via Action Units (AU) annotations, which describes in a continuous manifold the anatomical facial movements defining a human expression. However, such a prior for AU is not always available for all applications. Wu and Lu [2020] proposed an unsupervised disentanglement method, but it relies on a pretrained GAN that can translate any input facial image to a neural face. Huang and Yin [2022] combined Active Appearance Model (AAM) and cycleGAN [Zhu et al., 2017] for fine-grained facial expression generation. AAM can encapsulate the learned patterns of facial shape and texture variations associated with different expressions, which, however, requires careful image pre-processing and is computationally intensive.

Controllable DGMs based on Preferences

Two works applied GAN as a base for fine-grained I2I translation using relative attributes (RAs)[1], which refers to the preferences of image pairs on the strength of the interested attributes. The main differences lie in the strategies of incorporating the preference over the attributes into the image generation process. RCGAN [Saquil et al., 2018] adopted two critics consisting of a ranker, learning from the relative attributes of real images, and a discriminator, ensuring the image quality. Then the combination of two critics is aimed to guide the generator to produce high-quality fine-grained images while interpolating RAs on the generation. However, since the ranker is only trained with real images and generalizes poorly to generated images with interpolated RAs, the ranker would induce the generator to generate out-of-data manifold images which is opposite to the target of the discriminator, resulting in poor-quality images. RelGAN [Wu et al., 2019b] applied a matching-aware discriminator [2], which learns the joint data distribution of the triplet constructed with a pair

[1]Note that "relative attribute" is a widely used name in the field of I2I translation for the preferences of image pairs on the strength of the interested attributes.

[2]The matching-aware discriminator is modeled as a binary classifier as original GAN [Goodfellow et al., 2014]. The difference is that the matching-aware discriminator distinguishes real triplets from fake triplets while the discriminator in original GAN distinguishes real images from fake images.

of images and a discrete numerical label (i.e., relative attribute). In order to enable a smooth translation, RelGAN further interpolated the RAs on generated images while enforcing a constraint of interpolation quality. However, the constraint destroys smooth interpolation.
Work 5: We develop a new framework for high-quality fine-grained image-to-image translation using relative attributes (preferences) in Chapter 6.

2.3 Learning from Human Preferences

Learning to rank [Lin, 2010] learns a ranking score function from human preferences, where the function can compute ranking scores based on input features which, in turn, induces a ranking. Existing work falls into three categories: pointwise, pairwise and listwise ranking. The pointwise methods [Nallapati, 2004] utilize the order information per item from all the input preferences to define the ranking function and transform ranking into regression on single objects. Since these methods require global ranking information to assign scores to each item, they cannot be applied to the crowdsourcing scenario where partial preferences are collected [Pan, 2019]. The pairwise learning to rank organizes their inputs as pairwise comparison between pairs of items. The methods usually transform ranking into classification on object pairs [Burges et al., 2005, Cao et al., 2006], namely, predicting the preference of one item over another, e.g., $+1$ label for "better" or -1 label for "worse". The listwise methods model partial ranking lists, which directly maximizes consistency between the predicted ranking and the ground-truth ranking [Bruch et al., 2020]. The typical listwise ranking is ListNet [Cao et al., 2007b], whose loss function is defined as cross entropy between two parameterized probability distributions of permutations. one distribution is calculated via the predicted scores and the other is via the ground truth.

Ordinal regression [Gutiérrez et al., 2015] is a relevant branch in which all samples are required to be ranked on the same scale and are associated with an ordinal label. In contrast, preference learning involves comparing a list of partial samples from the dataset, and the rankings only need to be consistent within the list. Therefore, preference learning is more flexible and applicable to real-world scenarios.

2.4 Summary

In this chapter, we motivate our preliminary research on DGMs. Then, we discuss issues of existing work on desired data generation and highlight the significance of our main work, which is summarized as follows.

Issues of dataset level desired data generation and our solutions. Current methods all resort to complete knowledge about the property to select the desired data in order to learn the desired data distribution. Such knowledge is expensive and not in line with many real situations. On the other hand, all methods need to label the entire training data, which incurs huge costs. In this thesis, we propose learning the desired data distribution by human preferences, which only requires partial knowledge about the properties and can reduce the burden of labeling the whole training data [Pan et al., 2018, 2022, Liu, 2009]. Therefore, our work has the advantage of utilizing less and more accessible supervision than existing approaches (see Section 4.7 for empirical support).

Issues of desired data generation at the instance level and our solutions. While most existing methods resort to binary attributes, their fine-grained I2I translation is unsatisfactory as the binary attributes have restrictive description capacity. Some GAN-based works begun to conduct fine-grained I2I translation via RAs (preferences) but suffer from conflicting two goals: the goal of interpolating RAs on generated images for fine-grained translation and the goal for good-quality generation. We propose a new GAN framework for I2I translation using RAs in this thesis, which is the first GAN framework that reconciles the goal for fine-grained translation and the goal for high-quality generation.

Chapter 3

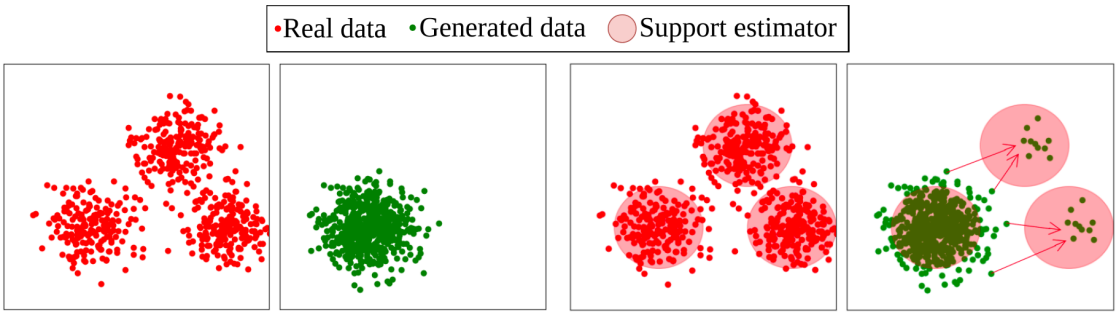
DGMs with Diversity and Disentanglement: A Preliminary Study

Our frameworks for generative modeling from human preferences are mainly built on GANs. Thus, to better understand GAN, we conduct a preliminary study on it, which targets solving its one of significant issues, mode collapse (Section 3.1). In addition, we study that a conditional DGM has better potential on downstream tasks than vanilla DGMs, which verifies the application prospect of controllable generation (arguably a broader topic to which this thesis belongs). Particularly, a disentangled VAE framework is proposed to improve clustering results on complex data contaminated by the unwanted factor (Section 3.2).

3.1 DGM with Mode Diversity

GAN is a powerful generative model known for its capability of generating sharp images. However, it is poor at generating diverse data, which refers to the mode collapse issue. Many works were proposed to solve it. In all, they can be summarized into two types: one is to make changes in the paradigm of the vanilla GAN [Hoang et al., 2018, Park et al., 2018, Gurumurthy et al., 2017, Lin et al., 2018, Salimans et al., 2016, Arjovsky et al., 2017a]; the other is to add an extra regularization term [Che et al., 2017, Du et al., 2018, Li et al., 2018].

In terms of the first type, some works increased the number of generators [Hoang et al., 2018, Park et al., 2018] motivated by forcing one generator to cover one different subset of the data. Gurumurthy et al. [2017] changed the simple latent distribution into a mixture of Gaussian model in order to increase the modeling power of the prior distribution. Lin et al. [2018] modified the discriminator to distinguish among multiple samples to improve mode diversity for GANs, similar motivation to mini-batch discrimination [Salimans et al., 2016].



(a) GAN with mode collapse (b) SR-GAN

Figure 3.1 (a) Problem: GAN suffers from the mode collpase, i.e., only generating one out of three modes. (b) Our solution (SR-GAN): red panels are the estimated support of real data. They are used to penalize the generated modes with no/scarce samples and guide the generator to disperse samples with all modes.

Some changed the original divergence to a new divergence metric, like Wasserstein distance [Arjovsky et al., 2017a].

The second type is to add an explicit regularization in the GAN's objective to improve mode diversity, which also motivates our work. In particular, there are two works most related to ours [Li et al., 2018, Du et al., 2018]. DAN-S [Li et al., 2018] used Maximum Mean Discrepancy to tell the difference between the real data distribution and the generated data distribution and used it as a regularization for GAN while LBT-GAN [Du et al., 2018] utilized the likelihood of the real data to guide the generator to cover all modes of the data through the density estimator. These two works are based on the statistics of the real data. However, LBT-GAN defines a bilevel optimization problem and has a high computational cost. DAN-S needs to define one more discriminator to discriminate among multiple samples.

Our idea to address mode collapse is also based on the estimation of real data statistics. But differently, we propose to estimate the support of the real data distribution, namely, capturing the regions in data space where the probability density lives. In doing so, we solve an easier problem than density estimation.

3.1.1 Support Matching as a Mode Regularizer

In this section, an extra regularization is introduced to improve mode diversity. In Fig. 3.1a, we can observe that when mode collapse arises, the support of the real data distribution, namely, the domain of data space, and the support of the generated data distribution cannot be matched, which motivates us to use such consistency as a regularization for GANs (Fig. 3.1b).

Support Estimation on Real Data Distribution

The support of a probability distribution refers to the regions in data space where the probability density is larger than zero [Schölkopf et al., 2001]. Support Vector Data Description (SVDD) [Tax and Duin, 2004] is one of the methods to estimate the support. It finds a minimum hypersphere to enclose all the input data and regards sample points outside the sphere as outliers of the data. The model is defined as follows:

$$\begin{aligned} &\min_{c,R,\boldsymbol{\xi}\geq 0} R^2 + \eta \sum_{i=1}^{\ell} \xi_i \\ &\text{s.t.} \left\|c - \phi\left(x_i\right)\right\|^2 \leq R^2 + \xi_i,\ \forall i = 1,\ldots,N, \end{aligned} \tag{3.1}$$

where R is the radius of the sphere. η is a nonnegative constant, which gives the trade-off between the two error terms: volume of the sphere and the number of rejected objects. c is the center of the sphere. ξ_i is slack variable for the training sample x_i. N is the number of the training samples. ϕ denotes a feature map to map input instances into the feature space.

ClusterSVDD In our method, we use ClusterSVDD [Görnitz et al., 2018] to capture the sub-structures of the support. This method unifies SVDD and k-means clustering, which fits K hyperspheres that can be defined by its centers and radius $\{c_k, R_k\}_{k=1}^K$, on the support of the real data distribution. With $\{c_k, R_k\}_{k=1}^K$, we calculate the cluster label y_i for each sample and collect samples with regard to each cluster $\mathcal{X}_k$ as follows:

$$y_i = \text{argmin}_{k\in\{1,\ldots,K\}} \left\|c_k - \phi\left(x_i\right)\right\|^2 - R_k^2,\ \forall i = 1,\ldots,N, \tag{3.2}$$

$$\mathcal{X}_k = \left\{x_i | y_i = k, i = 1,\ldots,N\right\},\ \forall k = 1,\ldots,K. \tag{3.3}$$

Each $\mathcal{X}_k$ is then used to solve one SVDD optimization problem (3.1). The procedure is repeated until convergence.

Support Matching as a regularizer

We use support matching between the real data distribution and the generated data distribution as a mode regularizer. With estimation on the support of the real data distribution, each sphere in ClusterSVDD will cover one sub-structure of the support, which can be also referred to one mode of the data. Therefore, the estimated spheres can be regarded as mode indicators for the generator to tell whether there is data generating in a certain mode.

We use the estimated spheres $\{c_k, R_k\}_{k=1}^K$ to evaluate the support of the generated data distribution and align it with that of the real data distribution. Specifically, we use Eq. (3.2)

to divide generated data into K groups $\{\mathcal{X}_k\}_{k=1}^{K}$ and match the size of the groups between the real data and the generated data. K is set to the number of the modes in the data. If some mode is missed by the generator, the size of its corresponding group would be zero, i.e., $|\mathcal{X}_k| = 0$, which causes the difference between the real data and the generated data. Such difference will guide the generator to generate data that is not covered currently. Since argmin function has no derivative, we instead replace it with softmax function for calculating the regularization term in GAN's objective as follows:

$$f_k(x_i) = \frac{\exp(-\beta(\|c_k - \phi(x_i)\|^2 - R_k^2))}{\sum_j \exp(-\beta(\|c_j - \phi(x_i)\|^2 - R_j^2))}. \tag{3.4}$$

The matching of the support between the real data distribution and the generated data distribution is regarded as a regularization for GAN. In short, the objective of SR-GAN consists of two terms, i.e., the discriminator and the generator:

$$\min_D \mathbb{E}_{P_r(x)}[-\log D(x)] + \mathbb{E}_{P(z)}[-\log(1 - D(G(z)))], \tag{3.5a}$$

$$\min_G \mathbb{E}_{P(z)}[-\log D(G(z))] + \lambda * \sum_k (\mathbb{E}_{P_r(x)}[f_k(x)] - \mathbb{E}_{P(z)}[f_k(G(z))])^2, \tag{3.5b}$$

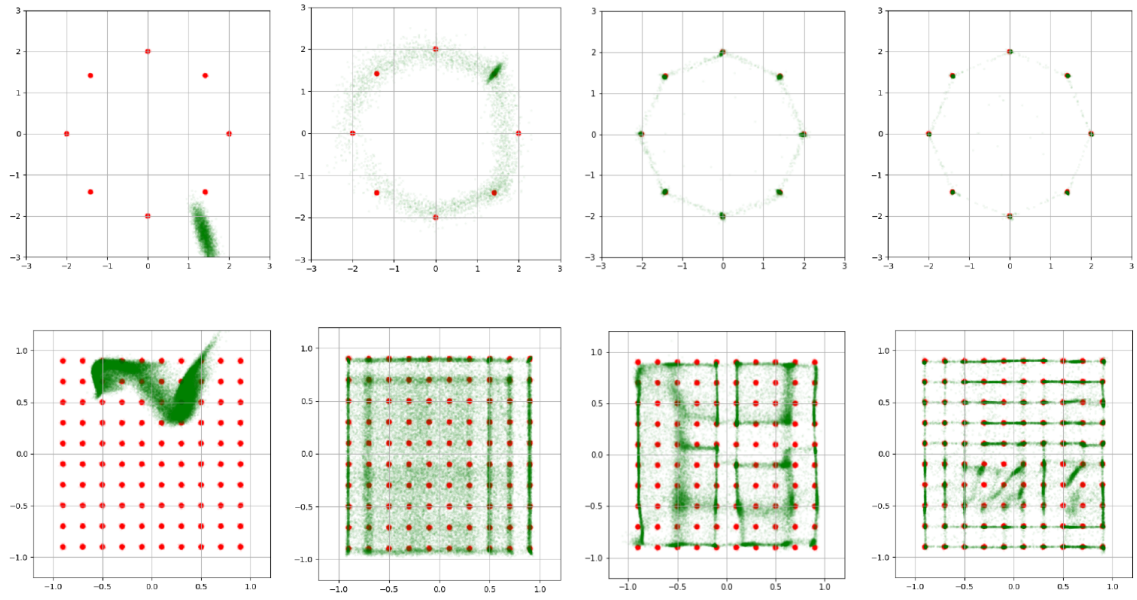
where G is denoted as the generator network. D is denoted as the discriminator network. $P(z)$ is the distribution of the input noise. $P_r(x)$ is the distribution of the real data. λ balances the image quality and the mode diversity.

3.1.2 Experiments

We apply our proposed SR-GAN on synthetic datasets and real-world datasets to evaluate the performance of SR-GAN in terms of improving mode diversity.

Baselines We compare SR-GAN with the vanilla GAN [Goodfellow et al., 2014], LBT-GAN [Du et al., 2018] and DAN-S [Li et al., 2018]. LBT-GAN and DAN-S are similar methods to ours, i.e., defining a regularization for GAN based on the statistics of the real data distribution.

Model Architectures & **Hyperparameters** Following [Metz et al., 2017], fully connected networks (FCNs) are used for the generator network and the discriminator network on the synthetic datasets and the SatImage dataset. A recurrent neural network (RNN) is used for the generator and a convolutional neural network (CNN) for the discriminator (RNN-CNN) on the MNIST dataset. Furthermore, the FCNs are also applied for the generator and the



(a) GAN (b) LBT-GAN (c) DAN-S (d) SR-GAN

Figure 3.2 Visual comparison of generated samples on the 2D ring data (Upper) and the 2D grid data (Lower). More overlapping between the generated samples and the real samples denotes a better generation.

discriminator (FCN-FCN) on MNIST following [Li et al., 2018]. We keep the architectures similar for all GANs in order to make a fair comparison.

In terms of the hyperparameters, the number of clusters K in CSVDD is set to the number of classes in the datasets. The trade-off factor λ is set to 1 for synthetic datasets and the SatImage dataset, 10 for MNIST with the RNN-CNN architecture and 50 for MNIST with the FCN-FCN architecture. β is set to 10 for all datasets. See Appendix for more details.

Synthetic Datasets

We construct two synthetic datasets following [Du et al., 2018]: (1) 2D ring, i.e., mixture of eight 2D Gaussian distributions with covariance matrix $0.02I$ arranged in a ring; (2) 2D grid, i.e., mixture of 100 2D Gaussian distributions with covariance matrix $0.01I$ arranged in a 10-by-10 grid.

To quantify the mode collapsing scenario, we adopt the following two metrics to measure the quality and diversity of the generated data.

Percentage of High Quality Samples (PHQS) [Srivastava et al., 2017] A sample is counted as a high quality sample if it is within three standard deviations of its nearest mode.

Table 3.1 Data quality (measured by PHQS), and mode diversity (measured by NMC) on the 2D ring and the 2D grid data. The results are averaged over five trials, with the standard error reported. Higher is better for both two metrics.

	2D Ring		2D Grid	
	PHQS (%)	NMC (Max 8)	PHQS (%)	NMC (Max 100)
GAN	0.2 ± 0.14	0.4 ± 0.24	7.7 ± 1.46	9.4 ± 1.69
LBT-GAN	10.4 ± 3.82	7.8 ± 0.20	14.5 ± 2.70	$\mathbf{100.0 \pm 0.00}$
DAN-S	47.5 ± 5.46	$\mathbf{8.0 \pm 0.00}$	15.7 ± 0.44	99.4 ± 0.24
SE-GAN	$\mathbf{91.7 \pm 0.97}$	$\mathbf{8.0 \pm 0.00}$	$\mathbf{45.3 \pm 2.46}$	$\mathbf{100.0 \pm 0.00}$

We regard one Gaussian component as one mode. Therefore PHQS can be regarded as an indicator for the quality of generated samples. Let l_i^* be the index of the nearest mode for the i-th sample:

$$l_i^* = \text{argmin}_{l \in \{1,\ldots,L\}} d(G(z_i), \mu_l), \ \forall i = 1, \ldots, N, \tag{3.6}$$

where μ_l is the mean of the l-th Gaussian component. L is the number of the components in the Gaussian mixture. d is a distance metric between the generated sample and the mean of one mode. PHQS is defined as follows:

$$PHQS = \frac{1}{N} \sum_{l=1}^{L} HQS(\mu_l) = \frac{1}{N} \sum_{l=1}^{L} \sum_{i=1, l_i^*=l}^{N} \mathbb{I}(d(G(z_j), \mu_l) < \varepsilon), \tag{3.7}$$

where HQS denotes high quality samples. $\mathbb{I}$ is an indicator function. ε is set to three standard deviations in the experiment.

Number of Mode Covered (NMC) [Srivastava et al., 2017] We use NMC to measure the mode diversity. A mode is counted as a covered mode if the generator creates at least one high quality samples of the mode. The NMC is defined as follows:

$$NMC = \sum_{l=1}^{L} \mathbb{I}(HQS(\mu_l) \geq 1). \tag{3.8}$$

Fig. 3.2 shows the visualization of samples generated by GAN and its variants on the 2D ring and the 2D grid datasets, respectively. We can observe that: (1) the vanilla GAN suffers from severe mode collapse problems on both datasets. Regarding the 2D ring data (Fig. 3.2 Upper), GAN only generates samples near one mode. Regarding the 2D grid data (Fig. 3.2 Lower), GAN covers few modes. (2) In terms of LBT-GAN, DAN-S and SR-GAN, all real samples are surrounded by the generated samples, which means that they can cover all modes of the data. (3) The density of the data generated by GAN's variants is not equal to that of the real data. However, SR-GAN can learn a closer distribution comparing to other baselines.

Table 3.2 Mode diversity (measured by NMC), and data quality (measured by KL) on SatImage. The results are averaged over five trials, with the standard error reported. Higher is better for NMC; lower is better for KL.

	NMC (Max 6)	KL
GAN	2.0 ± 0.77	1.38 ± 0.390
LBT-GAN	$\mathbf{6.0 \pm 0.00}$	0.21 ± 0.018
DAN-S	$\mathbf{6.0 \pm 0.00}$	0.06 ± 0.019
SR-GAN	$\mathbf{6.0 \pm 0.00}$	$\mathbf{0.02 \pm 0.005}$

Table 3.1 reports the evaluation using PHQS and NMC in terms of the data quality and the mode diversity, respectively. It shows that: (1) regarding the 2D ring data, GAN generates 0.2% PHQS and less than one mode in average. Regarding the 2D grid data, GAN generates 7.7% PHQS and around nine out of 100 modes. We thus conclude that the mode collapse indeed happens in the vanilla GAN. (2) However, LBT-GAN, DAN-S, and our SR-GAN do not encounter mode collapse problems on both two datasets. (3) In addition, our SR-GAN achieves the highest PHQS, which means that SR-GAN can generate more high quality samples than LBT-GAN and DAN-S.

SatImage Dataset

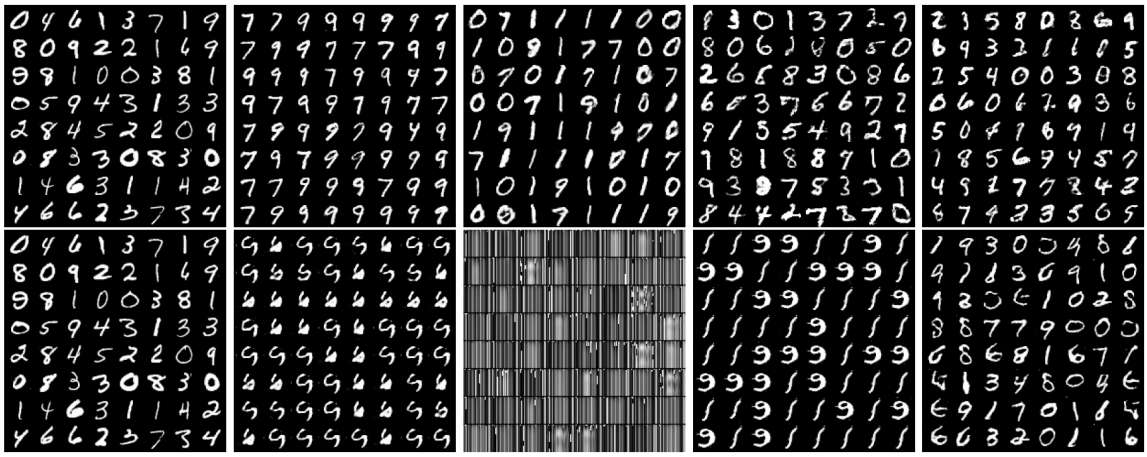
We then apply SR-GAN on a simple real-world dataset, i.e., SatImage dataset[1]. This dataset contains 4,435 instances with 36 attributes and 6 classes, where each class is regarded as one mode.

Number of Mode Covered (NMC) We also use NMC to evaluate the mode diversity on SatImage. The number of modes here is estimated using a trained classifier [Che et al., 2017]. We do not count high quality samples as above since it is hard to evaluate it on real-world datasets. Instead we count a mode as a covered mode if the number of its samples is greater than $\alpha\% \times \frac{\#samples}{\#modes}$ ($\alpha = 10$).

KL The KL divergence between the generated samples and the real samples over class [Metz et al., 2017] is used to evaluate the quality of the generated data.

It shows in Table 3.2 that (1) GAN only generates around two out of six modes, which denotes that GAN also suffers from a severe mode collapse problem on the simple real-world dataset. (2) LBT-GAN, DAN-S and our SR-GAN can cover all modes of the SatImage data. (3) In addition, our SR-GAN achieves the lowest KL divergence, which means that it can learn a more accurate data distribution.

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/multiclass/satimage.scale



(a) Real (b) GAN (c) LBT-GAN (d) DAN-S (e) SR-GAN

Figure 3.3 Visual comparison of generated samples on the MNIST data. **First column**: real samples from the MNIST dataset. **Upper column 2-5**: the generation with the FCN generator and the FCN discriminator. **Lower column 2-5**: the generation with the RNN generator and the CNN discriminator, which needs a more complex balance between the asymmetric architecture and thus is harder to train. The training of LBT-GAN with the RNN-CNN architecture is unstable and provides meaningless results.

MNIST Dataset

We further explore the superiority of our SR-GAN in terms of improving mode diversity on a more complex dataset: MNIST [Lecun et al., 1998]. It consists of 28×28 images with zero to nine digits. We adopt two architectures: FCN-FCN and RNN-CNN.

Instead of doing the support estimation on raw image data directly, we apply support estimation on the embedding space [Chalapathy and Chawla, 2019]. Particularly, we use deep neural networks [Hu et al., 2017] as feature extractors and input the discrete embedding features into ClusterSVDD. The dimension of the features is set to that of the input noise.

Evaluation We use NMC and KL introduced in Sect. 3.1.2 to measure the diversity and the quality of the generated data, respectively.

MNIST with the FCN-FCN architecture The upper panel of Fig. 3.3 shows that: (1) the vanilla GAN and LBT-GAN both suffer from a severe mode collapse issue on MNIST. The visualization shows that they only generate few of ten digits. (2) DAN-S and SR-GAN can significantly mitigate the mode collapse. However, the performance of SR-GAN is inferior to that of DAN-S. That is because, in SR-GAN, we train the support regularization independently from the GAN's objective for simplicity. A better result could be achieved

Table 3.3 Mode diversity (measured by NMC), and data quality (measured by KL) on MNIST with two architectures. The results are averaged over five trials, with the standard error reported. Higher is better for NMC; lower is better for KL. The results of LBT-GAN is unavailable since it generates meaningless samples.

	MNIST (FCN-FCN)		MNIST (CNN-RNN)	
	NMC (Max 10)	KL	NMC (Max 10)	KL
GAN	2.6 ± 0.25	1.57 ± 0.038	2.8 ± 0.80	1.70 ± 0.240
LBT-GAN	4.0 ± 0.00	1.28 ± 0.107	-	-
DAN-S	$\mathbf{10.0 \pm 0.00}$	$\mathbf{0.01 \pm 0.001}$	1.8 ± 0.20	1.86 ± 0.110
SR-GAN	8.2 ± 1.36	0.50 ± 0.259	$\mathbf{10.0 \pm 0.00}$	$\mathbf{0.12 \pm 0.025}$

through training them in a unified framework, which we leave for a future work. The results in Table 3.3 (Left) is consistent with the visualization results.

MNIST with the RNN-CNN architecture The RNN-CNN architecture is asymmetric, resulting in a more complex power balance [Metz et al., 2017]. Therefore, its training is much harder than the previous FCN-FCN architecture.

The lower panel of Fig. 3.3 shows that: (1) the samples generated by GAN, LBT-GAN and DAN-S with the RNN-CNN architecture are less sharper compared to the results with the FCN-FCN architecture. This is because the asymmetric RNN-CNN architecture is harder to train. LBT-GAN even generates meaningless samples. (2) The images generated by GAN and DAN-S share only one single style within one mode. (3) SR-GAN achieves the best performance and covers all modes. It can generate diverse styles for the digits. The results in Table 3.3 (Right) are consistent with the visualization.

3.1.3 Summary

In this part, we address the mode collapse problem by aligning the support of the generated data distribution with that of the real data distribution. The experiments show that our SR-GAN can avoid the mode collapse and also improve the data quality. SR-GAN introduces a simple extra regularization for GAN and does not modify any paradigm of GAN. Therefore the proposed support regularization term can be easy to be applied to other GANs, like conditional GAN, to solve the mode collapse problem.

3.2 Disentangled DGM for Cluster Analysis

Clustering is an essential technique for unsupervised data analysis, whose objective is to partition samples into groups so that the samples in the same group are similar while those from different groups are significantly different [Jain et al., 1999]. A desired clustering structure is expected to be semantically meaningful to humans [Xie et al., 2016]. In other words, a clustering algorithm should partition data in terms of the factor of interest [Jacob et al., 2016], e.g., the digital type of the MNIST data [Lecun et al., 1998].

Standard clustering [Cheng, 1995, Xie et al., 2016] is capable of capturing the desired semantic structures embedded in the clean raw data. However, in many real-world applications where the data commonly contains other variation factors, the discovery of the clustering structure in terms of the factor of interest would be adversely affected by these *unwanted factors* [Listgarten et al., 2010] (Fig. 3.4). While attaining interested features directly is infeasible in unsupervised clustering, undesired information is accessible in many practical scenarios, which can be removed so as to deliver more precise clustering results. For example, the variance in the source of data (e.g., different platforms for collecting images, which have different lighting conditions) may have a biased effect on the clustering structure, because the difference between clusters could be caused by the source of data instead of intrinsic semantic distinctions. Such information which indicates the unwanted factor can be obtained during data collection [Saenko et al., 2010]. Moreover, in some clustering tasks, domain experts can beforehand identify the unwanted factor that might cause spurious associations [Jacob et al.,

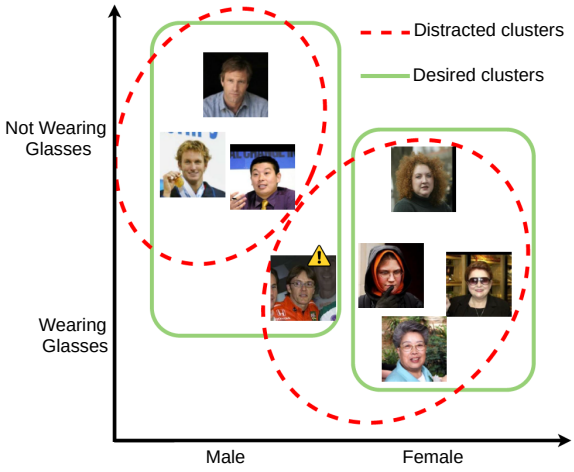


Figure 3.4 Motivation: raw images contain two important factors: gender and glass. Suppose clusters with green solid lines are the desired clustering results, where the partitions are based on the gender factor only. Standard clustering algorithms that neglect the unwanted factor obtain clusters distracted by the glass factor, denoted by red dash lines. Notably, the face (a man wearing glasses) with the warning sign is viewed incorrectly grouped.

2016]. Therefore, in this study, we perform clustering while removing the negative effect of the unwanted factor.

Previous methods [Jacob et al., 2016, Gagnon-Bartsch and Speed, 2012] simply projected raw data onto the subspace orthogonal to the space expanded by the unwanted factor under the linear assumption before clustering. Specifically, they decompose the data into linear combinations of the desired factor and the unwanted factor. In the linear space, they remove the unwanted information by simply subtracting the unwanted covariate from the data. In parallel, Benito et al. [Benito et al., 2004] applied an improved SVM which finds a linear hyperplane to separate two classes (i.e., the binary unwanted factor) in a supervised manner and then projects the raw data on this hyperplane. Such a method cannot scale to the scenario with multiple classes beyond the binary unwanted factor argued in [Johnson et al., 2007]. In summary, all previous approaches are limited to the raw feature space, which may not capture high-level and representative features to describe the interested factor as well as the unwanted factor. In addition, these methods only consider linear dependence between the data and the unwanted factor for clustering, which oversimplifies the real situations. The two flaws restrict existing methods from applying to complex real-world data, where both the factor of interest and the unwanted factor are non-linearly embedded in the raw data.

Other works potential for this task relate to two clustering branches. One is alternative clustering [Niu et al., 2013, Wu et al., 2018, 2019a], which finds an alternative clustering given a dataset and an existing pre-computed clustering. It maximizes the dependence between the clustering subspace and the alternative clustering partition while simultaneously minimizes the dependence between the subspace and the given clustering partition. Then the obtained clustering results can filter out the information of the unwanted clustering structure. However, these methods are restrictive to pre-defined kernel functions used for evaluating the dependence, which are not flexible and have limited description power. The methods only work well for low-dimensional and simple datasets. Another is fair clustering, particularly the most recent work [Li et al., 2020a] proposed to learn fair and clustering-favorable representations for clustering by filtering out unfair attributes from representations. In particular, it incorporated the advance of representation learning in deep learning into clustering framework. However, their model separately learns the deep representation module and the clustering module, which requires a careful balance between these two modules. In addition, the adversarial training designed for the removal of the unfair factor also increases the difficulty of model training.

In the following, we present a new clustering framework called COUF, which removes the unwanted factor in the semantic latent space of complex data through a non-linear dependence measure during clustering.

3.2.1 Clustering without the Unwanted Factor (COUF)

Definition 1 (Clustering without the unwanted factor)**.** *Let $\mathcal{X} \in \mathbb{R}^{N\times U}$ be a dataset with N samples and U features. Let $\mathcal{C} = [c_1, c_2, \ldots, c_N]^T \in \{0,1\}^{N\times V}$ be the corresponding labels with regards to a certain unwanted factor c, where $\mathcal{C}_{i,j} = 1$ if x_i belongs to class j and $\mathcal{C}_{i,j} = 0$ otherwise; V is the number of categories. Our goal is to find a partition $\Omega_x \in \Pi_{K,x}$, such that Ω_x is uninformative of c. The objective is formulated as:*

$$\min_{\Omega_x \in \Pi_{K,x}} F(\Omega_x), \quad s.t.\ \Omega_x \perp c, \tag{3.9}$$

where $\perp$ denotes that two variables are independent. $\Pi_{K,x}$ denotes all feasible K-partitions of $\mathcal{X}$[2]. Ω_x represents a K-partition in the space where x is located. F is the clustering objective, whose minimization aims at optimizing the quality of clustering. For instance, the k-means clustering objective is $F = \sum_{k=1}^{K}\sum_{n=1}^{N} \omega_{nk}\|x_n - e_k\|_2^2$, where e_k is the k-th cluster centroid. $\omega_{nk} \in \{0,1\}$ denotes the cluster assignment which equals 1 if x_n is assigned to the k-th cluster and 0 otherwise.

Deep semantic clustering in the latent space

Representation learning can output high-level and abstract representations [Vincent et al., 2010]. Thus, we perform clustering in the latent space to capture the semantic structure of complex data. To be specific, we jointly learn latent representations and clustering within a deep neural network (DNN). The clustering loss is built over the representations extracted from DNNs [Xie et al., 2016, Niu et al., 2022].

Let's consider a general task (e.g., data reconstruction) that involves encoding the data x into its latent representation z using the posterior $Q(z \mid x)$, which serves as an encoder. The objective of deep semantic clustering includes the objective L for representation learning and the objective F for clustering on the representations. Namely,

$$\min_{q,\ \Omega_z \in \Pi_{K,x}} L(Q,x) + \eta F(\Omega_z). \tag{3.10}$$

Ω_z denotes a partition in the space where z resides. $\Pi_{K,x}$ is defined similarly as $\Pi_{K,x}$ in Eq. (3.9). η is a trade-off parameter that balances representation learning and clustering.

In particular, we choose Variational AutoEncoder (VAE) [Kingma and Welling, 2014] to compute $L(Q,x)$, because VAE includes modeling of $Q(z \mid x)$, and VAE based clustering

[2] A K-partition of a set $\mathcal{X}$ denotes a collection of K mutually disjoint non-empty subsets whose union is X. Namely, $\Omega_x = (\Omega_1, \Omega_2, \ldots, \Omega_K)$, where $\bigcup_{i=1}^{K} \Omega_i = X, \Omega_i \cap \Omega_j = \emptyset, 1 \le i \ne j \le K$.

can obtain good clustering-favorable representations and is effective for various complex datasets [Jiang et al., 2017].

Clustering with representations invariant to unwanted factor

Eq. (3.10) conducts semantic clustering without considering the existence of the unwanted factor c. To eliminate the negative impact of c on the target clustering structure Ω_z, we propose deep semantic clustering without the unwanted factor. Recalling Eq. (3.9), our objective is formulated as:

$$\begin{aligned} &\min_{q,\, \Omega_z \in \Pi_{K,x}} L(Q,x) + \eta F(\Omega_z), \\ &s.t.\ \Omega_z \perp c. \end{aligned} \tag{3.11}$$

Since a partition Ω_z is defined over the whole dataset while c is collected per sample, directly implementing $\Omega_z \perp c$ is complex and incurs large computational costs. Instead, we impose an alternative independence constraint between the sample representation z and the unwanted factor c, i.e., $z \perp c$, both of which are defined at the sample level.

Proposition 1. *Let $\mathcal{Z} = \{z_1, z_2, \ldots, z_N\}^T \in \mathcal{Z}$ be the representation set of the dataset $\mathcal{X}$. Suppose the clustering algorithm $\mathcal{A}$ takes $\mathcal{Z}$ as an input and returns a partition Ω_z of $\mathcal{Z}$. Namely, $\mathcal{A} : \mathcal{Z} \longrightarrow \Omega_z$. If $z \perp c$, then we naturally have $\Omega_z \perp c$.*

Proposition 1 demonstrates clustering over representations z that is invariant to the unwanted factor c can derive a clustering structure Ω_z that is uninformative of the unwanted factor c. Thus, our objective can be reformulated as:

$$\begin{aligned} &\min_{q,\, \Omega_z \in \Pi_{K,x}} L(Q,x) + \eta F(\Omega_z), \\ &s.t.\ z \perp c. \end{aligned} \tag{3.12}$$

The independence constraint $z \perp c$ is still a strong condition and is difficult to be optimized directly. We apply the minimization of the mutual information $I(z,c)$ to approximate it [Moyer et al., 2018]. Adding the term $I(z,c)$, the objective Eq. (3.12) is further formulated as:

$$\min_{q,\, \Omega_z \in \Pi_{K,x}} L(Q,x) + \eta_1 I(z,c) + \eta_2 F(\Omega_z). \tag{3.13}$$

where η_1 and η_2 are the hyper-parameters that balance the three losses. In Eq. (3.13), the interested clustering factor, which is embedded in the representation z, and the unwanted factor c can be semantically described in the latent space [Xie et al., 2016, Vincent et al., 2010]. Meanwhile, these two factors are disentangled in the latent space. By optimizing Eq. (3.13), we can obtain a semantic clustering structure Ω_z that is irrelevant to the unwanted features.

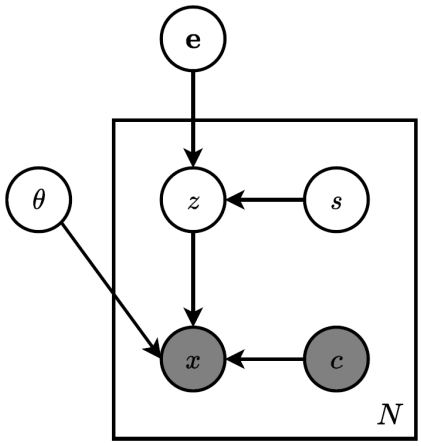


Figure 3.5 Graphical model of our COUF. x and c denote an observed sample and the corresponding observed unwanted factor, respectively. $\mathbf{e} = \{e_1, e_2, \ldots, e_K\}$ denotes clustering centroids. A latent representation z is generated based on a clustering centroid picked from $\mathbf{e}$ according to the group assignment ω. θ is the parameter of a decoder which decodes z and c into x. The rectangle is "plate notation", which means that we can sample x, c, z, and ω for N times while θ and $\mathbf{e}$ remain fixed.

The overall clustering framework

To summarize, our framework jointly trains with three modules. First, the VAE structure is adopted as the feature extractor module for learning semantic features. Further, we introduce one disentangling module (Fig. 3.6) over the latent space derived by VAE, to disentangle the unwanted factor c and other salient information z encoded in the data (i.e., $z \perp c$). Last, a clustering module (Fig. 3.7) based on the soft k-means is incorporated within the VAE structure to perform clustering on the factor of interest that is embedded in z only.

Variational autoencoder According to our graph model (Fig. 3.5), we formulate the statistical (non-linear) dependence between x and c in the latent space. Namely,

$$P(x,z,c) = P(z,c)P(x \mid z,c),$$

where z is the latent variable of x.

Similar to VAE [Kingma and Welling, 2014], the variational lower bound for the expectation of conditional log-likelihood $\mathbb{E}_{(x,c)}[\log P(x \mid c)]$ can be deduced as follows:

$$\mathbb{E}_{(x,c)}[\log P(x \mid c)] \geq \mathbb{E}_{(x,c)}\left[\mathbb{E}_{z\sim Q(z|x)}[\log P(x \mid z,c)] \; - KL[Q(z \mid x)\|P(z)]\right]. \tag{3.14}$$

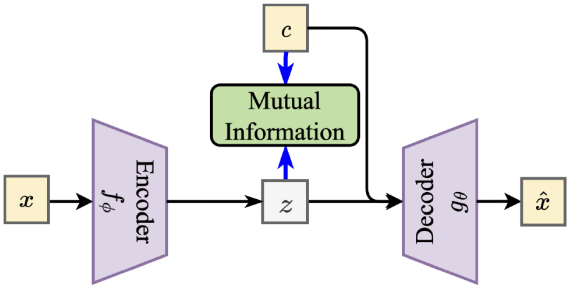


Figure 3.6 Disentanglement via mutual information minimization.

The conditional decoder $P(x \mid z,c)$ takes both z and c as input. We simplify the distribution of z to only depend on the input x, optimized by the encoder $Q(z \mid x)$. $P(z)$ is the prior distribution which is defined as a Gaussian noise.

Following VAE, we parameterize the approximate posterior $Q(z \mid x)$ with an inference network (encoder) f_ϕ that encodes a data sample x to its latent embedding z, and parameterize the likelihood $P(x \mid z,c)$ with a conditional decoder g_θ that produces a data sample conditioned both on the latent embedding z and the observed factor c (Fig. 3.5). Usually, a particle z_n is sampled from $Q(z \mid x)$ for reconstructing x_n [Kingma and Welling, 2014]. Then, the loss function (minimization) based on the Monte Carlo estimation of the variational lower bound in Eq. (3.14) is defined as:

$$\mathcal{L}_{\text{VAE}} = \sum_{n=1}^{N} \ell_{\text{r}}(x_n, g_\theta(z_n, c_n)) + \sum_{n=1}^{N} KL[Q_\phi(z \mid x_n) \| P(z)], \tag{3.15}$$

where ℓ_{r} denotes the reconstruction loss, which can be instantiated with, e.g., mean squared loss or cross-entropy loss. $\mathcal{L}_{\text{VAE}}$ is used to calculate the first term $L(Q,x)$ in Eq. (3.13).

Disentanglement by minimizing mutual information By minimizing the mutual information $I(z,c)$ between the latent variable z and the unwanted factor c, the unwanted factor is disentangled from other salient information in the latent space.

Lemma 1 (MI upper bound [Moyer et al., 2018])**.** *The mutual information between the latent representation z and the unwanted factor c, i.e., $I(z,c)$, is subject to a variational upper bound:*

$$I(z,c) \leq -H(x \mid c) - \mathbb{E}_{x,c,z\sim q}[\log P(x \mid z,c)] + \mathbb{E}_x[KL[Q(z \mid x) \| Q(z)]]. \tag{3.16}$$

As $I(z,c)$ is not directly computable, we instead use its upper bound Eq. (3.16). The constant $H(x \mid c)$ will be ignored. The third term on the right of Eq. (3.16) is hard to calculate since it contains the empirical marginal distribution $Q(z)$. We follow Moyer et al. [2018] to

approximate it using the pairwise distances:

$$KL[Q(z \mid x) \| Q(z)] \approx \sum_{x} \sum_{x'} KL\left[Q(z \mid x) \| Q(z \mid x')\right].$$

By framing $Q(z \mid x)$ and $P(x \mid z, c))$ as VAE's encoder f_ϕ and decoder g_θ, respectively, the loss function is finally defined as:

$$\mathcal{L}_{\text{MI}} = \sum_{n=1}^{N} \ell_{\text{r}}\left(x_n, g_\theta(z_n, c_n)\right) + \sum_{n=1}^{N} \sum_{m=1}^{N} KL[Q_\phi(z \mid x_n) \| Q_\phi(z \mid x'_m)] \tag{3.17}$$

The minimization of $I(z,c)$, the second term in Eq. (3.13), is thus replaced by the minimization of its upper bound, i.e., $\mathcal{L}_{\text{MI}}$.

Clustering over the c-invariant embedding Eq. (3.17) enables us to filter out the information of the unwanted factor c from the latent code z. For the sake of efficiency, we apply k-means algorithm to conduct clustering on the c-invariant embedding z. Note that k-means is a limit case of the Expectation-Maximization algorithm for Gaussian mixture model [Bishop, 2006]. Thus, the latent embedding z can be roughly said generated with the following process [Jiang et al., 2017] (Fig. 3.5): (1) a clustering centroid e_k is picked from $\mathbf{e}$ according to the corresponding clustering assignment ω; (2) a latent embedding z is generated based on e_k.

Particularly, the k-means clustering loss is defined as:

$$\mathcal{L}_{\text{cluster}} = \sum_{n=1}^{N} \sum_{k=1}^{K} \omega_{nk} \left\| z_n - e_k \right\|_2^2. \tag{3.18}$$

$\mathcal{L}_{\text{cluster}}$ is used to compute the third term $F(\Omega_z)$ in Eq. (3.13). $\mathbf{e} = \{e_1, e_2, \ldots, e_K\}$ are the collection of K centroids. $\omega_{nk} \in \{0,1\}$ refers to the group assignment that assigns the latent embedding z to its closest clustering centroid. Namely,

$$\lambda_{nk} = \frac{\exp\left(-\tau \left\| z_n - e_k \right\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(-\tau \left\| z_n - e_i \right\|_2^2\right)}, \tag{3.19a}$$

$$\omega_{nk} = \begin{cases} 1 & k = \operatorname{argmax}_j \lambda_{nj} \\ 0 & \text{otherwise} \end{cases}, \tag{3.19b}$$

where $k = 1, 2, \ldots, K$. τ is the temperature and is set to 5 in the experiment.

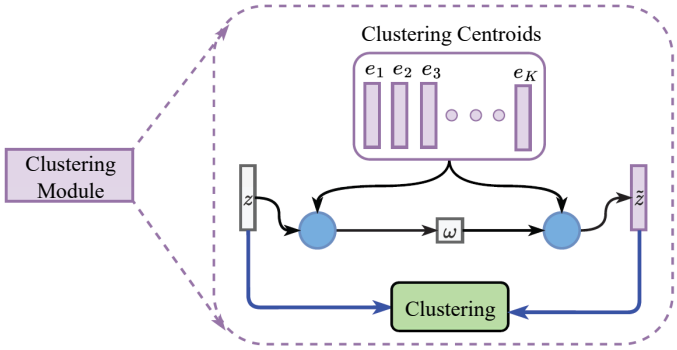


Figure 3.7 Clustering over c-invariant embedding z. The first blue circle means that z and $\mathbf{e}$ determine ω according to Eq. (3.19) while the second one means that ω and $\mathbf{e}$ determine $\tilde{z}$ according to Eq. (3.20).

Due to the reconstruction loss in VAE (Eq. (3.15)), the latent representations would contain many sample-specific details, which is detrimental to clustering. We follow [Pan and Tsang, 2021] to introduce the following skip-connection formulation to unify the reconstruction goal and the clustering goal. Namely,

$$\hat{z}_n = h_\psi\left(z_n, \tilde{z}_n\right), \text{ where } \tilde{z}_n = \sum_{k=1}^{K} \omega_{nk} e_k. \tag{3.20}$$

Note that $\tilde{z}_n$ is one of K clustering centroids as ω_{nk} is a one-hot assignment. h_ψ constructs a new latent representation $\hat{z}_n$ that incorporates not only the original c-invariant embedding z_n but also its belonging clustering centroid $\tilde{z}_n$ as the input of the decoder (Fig. 3.8). h_ψ is implemented as a linear layer.

Our proposed framework: COUF Integrating all three modules comes to our Clustering withOut Unwanted Factor (COUF) (Fig. 3.8). The final objective of COUF is formulated as:

$$\mathcal{L}(\Theta, \mathbf{e}) = \mathcal{L}_{\text{VAE}} + \eta_1 \mathcal{L}_{\text{MI}} + \eta_2 \mathcal{L}_{\text{cluster}}, \tag{3.21}$$

where $\Theta = \{\theta, \phi, \psi\}$ denote the network parameters and $\mathbf{e}$ represent clustering parameters. η_1 and η_2 are the trade-off parameters.

Clustering structure. After training the model, the clustering structure $\Omega_z = (\Omega_1, \Omega_2, \ldots, \Omega_K)$ is calculated by:

$$\Omega_k = \{z_n \mid \omega_{nk} = 1, n = 1, 2, \ldots, N\},$$

where $k = 1, 2, \ldots, K$ and ω_{nk} is defined in Eq. (3.19b).

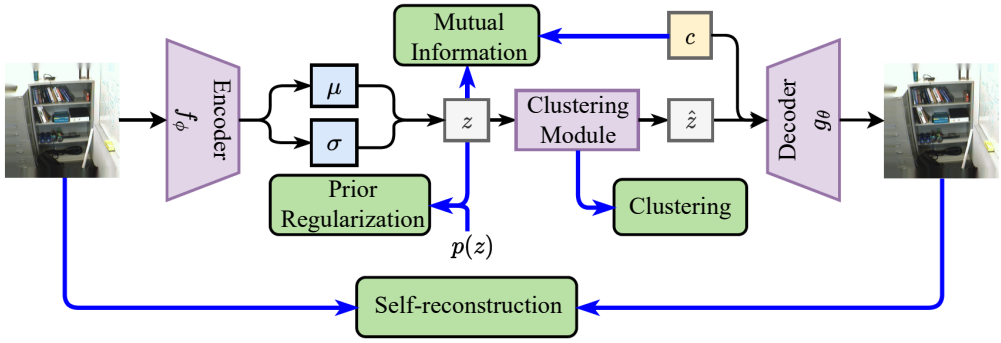


Figure 3.8 The architecture of our Clustering without the Unwanted Factor (COUF).

Alternative update with stochastic optimization In Eq. (3.21), two types of parameters, i.e., network parameters Θ, and clustering parameters $\mathbf{e}$, are coupled together, which hinders them from joint optimization. We adopt coordinate descent [Wright, 2015] to alternatively optimize Θ and $\mathbf{e}$. Fixing $\mathbf{e}$, the network parameters Θ is updated with gradient descent:

$$\Theta^{(t)} := \Theta^{(t-1)} - \kappa \frac{\partial \mathcal{L}\left(\Theta, \boldsymbol{e}^{(t-1)}\right)}{\partial \Theta}\bigg|_{\Theta=\Theta^{(t-1)}}, \tag{3.22}$$

where κ is the step size. t is the iteration index.

Fixing Θ, the clustering parameters $\mathbf{e}$ is updated with gradient descent:

$$\mathbf{e}^{(t)} := \mathbf{e}^{(t-1)} - \kappa \frac{\partial \mathcal{L}\left(\Theta^{(t)}, \mathbf{e}\right)}{\partial \mathbf{e}}\bigg|_{\mathbf{e}=\mathbf{e}^{(t-1)}}. \tag{3.23}$$

The gradient of clustering parameters can be calculated analytically.

To make our COUF scalable to large-scale problems, we apply mini-batch stochastic gradient updates for all parameters. However, such an update for clustering centroids $\mathbf{e}$ would be unstable because the clustering centroids estimated by different mini-batch data may be of great discrepancy. To overcome this limitation, we apply the exponential moving average (EMA) update for the centroids since the EMA update yields good stability [Van Den Oord et al., 2017, Kaiser et al., 2018]. Specifically, each centroid e_k is updated online using the

Algorithm 1 Clustering withOut the Unwanted Factor (COUF)

Input: Observed data $\{(x_n, c_n)\}_{n=1}^N$ with sample $x_n \in \mathbb{R}^d$ and the unwanted factor $c_n \in \{0,1\}^G$, network $\{f_\phi, g_\theta, h_\psi\}$, #centroid K, #epoch M, batch size B.
Output: network $\{f_\phi, g_\theta, h_\psi\}$, clustering centroids $\mathbf{e} = (e_1, e_2, \dots e_K)$, clustering assignment $\{\omega_{nk}\}$.
Initialize: network parameters $\{\phi, \theta, \psi\}$ and clustering centroids $\mathbf{e}$.
for epoch$= 1, 2, \dots, M$ **do**
  for iteration$= 1, 2, \dots, \lfloor \frac{N}{B} \rfloor$ **do**
    Sample a mini-batch of pairs $\{(x_n, c_n)\}_{n=1}^B$.
    Calculate clustering assignment $\{\omega_{nk}\}$ by Eq. (3.19).
    Update network parameters $\{\phi, \theta, \psi\}$ by Eq. (3.22).
    Update clustering centroids $\mathbf{e}$ using Eq. (3.24).
  end for
end for

assigned neighbor representations in the mini-batches $\{z_b\}_{b=1}^B$:

$$\begin{aligned} B_k^{(t)} &:= B_k^{(t-1)} \times \gamma + \sum_{b=1}^{B} \omega_{bk}^{(t-1)} \times (1-\gamma), \\ \mu_k^{(t)} &:= \mu_k^{(t-1)} \times \gamma + \sum_{b=1}^{B} \omega_{bk}^{(t-1)} \times z_b^{(t-1)} \times (1-\gamma), \\ e_k^{(t)} &:= \frac{\mu_k^{(t)}}{B_k^{(t)}}, \end{aligned} \tag{3.24}$$

where $\gamma \in [0,1]$ is a decay parameter, which is set to 0.995 for all datasets. The training algorithm is summarized in Algorithm 1.

Theoretical analysis

In this section, we theoretically analyze that optimizing network parameters Θ of COUF in Eq. (3.21) is equivalent to (1) maximizing the lower bound of the mutual information between the representation and the interested clustering structure, while (2) minimizing the upper bound of the mutual information between the representation and the unwanted factor.

Theorem 1. *Assume a fixed clustering structure, i.e., the clustering centroids* $\mathbf{e} = \{e_1, e_2, \dots, e_K\}$ *and the clustering assignments* $\{\omega_n\}_{n=1}^N$*, where* ω_n *is a K-dimensional one-hot vector and* ω_{nk} *is defined in Eq.* (3.19b)*. The minimization of our clustering object* $\mathcal{L}_{cluster}$ *is equivalent to maximizing the variational lower bound of the mutual information between the representa-*

tion z and the interested clustering structure, represented by the group assignment ω, i.e., $I(z,\omega)$.

The proof is put in Appendix A.

Corollary 1. *Let $\mathcal{L}'(\Theta)$ denote the objective of Eq.* (3.21) *for optimizing network parameters $\Theta = \{\theta, \phi, \psi\}$. Given the observed data pairs $\{(x_n, c_n)\}_{n=1}^N$, the clustering assignments $\{\omega_n\}_{n=1}^N$ and the centroids* **e**, *$\mathcal{L}'(\Theta)$ is subject to the following lower bound:*

$$\mathcal{L}'(\Theta) \geq -\mathbb{E}_{(x,c)}[\log P(x \mid c)] + \eta_1 I(z,c) - \eta_2 I(z,\omega). \tag{3.25}$$

From Corollary 1, we conclude that the optimization for Θ when fixing **e**, i.e., Eq. (3.22), is to learn a clustering-favorable representation, which is invariant to the unwanted factor.

3.2.2 Experiments

Dataset. We conduct experiments on five public high-dimensional image datasets, which have various numbers of clusters. Their statistics are summarized in Table 3.4. More details are desrbied in Appendix A.

Implementations. We employ a multi-layer perception (MLP) VAE architecture (described in the Appendix) for all datasets. Compared with those AE-based clustering methods [Xie et al., 2016, Guo et al., 2017], our COUF introduces only one extra linear layer for Eq. (3.20), which bring negligible network parameter overhead. We apply COUF to raw data for *UCI-Face*, *Rotated Fashion* and *MNIST-USPS* considering their simplicity. Inspired by the recent state-of-art (SOTA) clustering methods [Tsai et al., 2021, Niu et al., 2022], which rely on structured representations to achieve superior performance on complex datasets, we apply COUF to the extracted features for *Office-31* and *CIFAR10-C* considering their complexity. We use ImageNet-pretrained ResNet50 [He et al., 2016] to extract features for Office-31 following the SOTA clustering method on Office-31 [Li et al., 2020a]. We use MoCo [He et al., 2020] to extract features for *CIFAR10-C* following the SOTA clustering

Table 3.4 The statistics of datasets. The digit in the brace indicates the number of categories.

Dataset	#sample	#dim	#cluster (K)	unwanted factor (V)
UCI-Face	1,872	32×30	4	identity (20)
Rotated Fashion	30,000	28×28	5	cloth category (6)
MNIST-USPS	67,291	32×32	10	source of digit (2)
Office-31	3,612	$224 \times 224 \times 3$	31	domain source (2)
CIFAR10-C	40,000	$32 \times 32 \times 3$	10	corruption type (4)

method on *CIFAR10-C* [Niu et al., 2022]. Note that these feature extractors do not utilize any supervision regarding the datasets. Throughout all tasks, the dimension of centroids is set to 10 following [Xie et al., 2016, Guo et al., 2017]. The centroids are randomly initialized. We adopt the Adam optimizer. The default learning rate, training epoch, and batch size are 5e-4, $1,000$, and 256, respectively.

Baselines. The method that removes the unwanted factor in the raw space via linear projection, i.e., RUV [Jacob et al., 2016] is included as our first baseline. Further, we extend RUV to eliminate the unwanted factor in the latent space. In Particular, we first train AE to obtain the latent representations for *UCI-Face*, *Rotated Fashion*, and *MNIST-USPS*. We use the extracted features described above as the representations for *Office-31* and *CIFAR10-C*. Then, we apply RUV to remove unwanted information from the representations. We name these two baselines as RUV_x and RUV_z, respectively. We also consider Iterative Spectral Method (ISM) [Wu et al., 2019a]) and Deep Fair Clustering (DFC) [Li et al., 2020a] as our baselines since these two methods can be deemed as the same objective as ours (Eq. (3.9)). *For a fair comparison, we take raw images of UCI-Face, Rotated Fashion and MNIST-USPS and extracted features of Office-31 and CIFAR10-C as input for all the baselines except for* RUV_x*, which takes raw data as input.*

Metrics. We evaluate different clustering methods with two widely-used clustering metrics, i.e., accuracy (ACC) and normalized mutual information (NMI). For both two metrics, values range between 0 and 1, and a higher value indicates better performance.

Table 3.5 Comparison of various methods w.r.t. clustering validity, ACC (↑) and NMI (↑). The best results are highlighted in bold. The second-best results are underlined.

Dataset	Metric	ISM	DFC	RUV_x	RUV_z	COUF
UCI-Faces	ACC	0.763	0.394	0.380	0.539	**0.824**
	NMI	0.454	0.087	0.163	0.322	**0.570**
Rotated Fashion	ACC	N.A.	0.539	0.579	**0.993**	<u>0.985</u>
	NMI	N.A.	0.351	0.516	**0.969**	<u>0.940</u>
MNIST-USPS	ACC	N.A.	<u>0.825</u>	0.457	0.785	**0.919**
	NMI	N.A.	<u>0.789</u>	0.379	0.756	**0.837**
Office-31	ACC	0.659	<u>0.692</u>	0.186	0.673	**0.724**
	NMI	0.671	<u>0.718</u>	0.232	0.714	**0.728**
CIFAR10-C	ACC	N.A.	0.283	0.208	<u>0.357</u>	**0.458**
	NMI	N.A.	0.186	0.085	**0.317**	<u>0.311</u>

Performance comparison

The quantitative results of our COUF and various baselines on five datasets are summarized in Table 3.5. We observe that:

(1) **COUF is better than all baselines.** COUF obtains superior results on all datasets. This is because it adopts an effective non-linear dependence measure, i.e., mutual information, and jointly trains a representation module, a clustering module, and a disentanglement module, which can learn clustering-favorable representations invariant to the unwanted factor.

(2) **Latent space is better than raw space. Non-linear correlation is better than linear correlation.** RUV_z achieves better performance than RUV_x, which shows that removing the unwanted factor in the latent space is more effective than in the raw space of the image datasets. RUV_z obtains worse results than our COUF on the four datasets since RUV_z simply adopts linear projection and heavily relies on the extracted representations beforehand, which cannot deal with these complex datasets where the desired clustering factor and the unwanted factor are coupled non-trivially in the latent space. RUV_z performs slightly better than COUF on *Rotated Fashion*. It is probably because the rotation factor (desired) and the cloth category factor (unwanted) are linearly separable in the latent space.

(3) **DFC originally designed for two categories degenerates on the dataset with more categories** (i.e., *UCI-Faces*, *Rotated Fashion*, and *CIFAR10-C*). On one hand, more categories may increase the difficulty of adversarial training, making it unable to effectively remove the unwanted factor. On the other hand, the constraint to maintaining the clustering structure in DFC is harder to satisfy in this situation. Specifically, the constraint requires training a DEC [Xie et al., 2016] for each category of data. For example, it needs to train a DEC on around 93 images for *UCI-Face*, which would suffer from insufficient training samples.

(4) **ISM cannot be executed on large-scale datasets**, i.e, *Rotated Fashion*, *MNIST-USPS* and *CIFAR10-C* because it performs clustering with full batch data training. Specifically, it requires a memory complexity of $\mathcal{O}(n^2)$ and needs to store a data matrix with a size larger than $10k \times 10k$ for these datasets, which is beyond our computing capacity. On the small-scale datasets, i.e., UCI-Faces and Office-31, ISM is still inferior to our COUF.

Efficacy of removing the unwanted factor for clustering

To demonstrate the gain of clustering that takes into account the removal of the unwanted factor, we include the comparison with standard clustering methods, which neglects the unwanted factor and performs clustering directly on raw data or data representations, namely,

Table 3.6 COUF compared with standard clustering w.r.t. clustering validity, ACC (↑) and NMI (↑) on three simple image datasets.

Dataset	Metric	k-means	IDEC	COUF
UCI-Faces	ACC	0.266	0.356	**0.824**
	NMI	0.002	0.069	**0.570**
Rotated Fashion	ACC	0.487	0.602	**0.985**
	NMI	0.414	0.611	**0.940**
MNIST-USPS	ACC	0.506	0.789	**0.919**
	NMI	0.447	0.766	**0.837**

k-means [Bishop, 2006], IDEC [Guo et al., 2017][3], PICA [Huang et al., 2020] and SPICE [Niu et al., 2022][4]. We apply PICA and SPICE only on *Office-31* and *CIFAR10-C* considering that they were proposed for complex image datasets. For a fair comparison, we take raw images of *UCI-Face*, *Rotated Fashion* and *MNIST-USPS* and extracted features of *Office-31* and *CIFAR10-C* as input for the methods except for PICA. PICA takes raw images of all datasets as input since it needs to conduct image augmentations for partition confidence maximization [Huang et al., 2020].

Table 3.7 COUF compared with standard clustering w.r.t. clustering validity, ACC (↑) and NMI (↑) on two complex image datasets.

Dataset	Metric	k-means	IDEC	PICA	SPICE	COUF
Office-31	ACC	0.648	0.634	0.440	0.231	**0.724**
	NMI	0.689	0.690	0.536	0.341	**0.728**
CIFAR10-C	ACC	0.247	0.420	0.220	0.313	**0.458**
	NMI	0.225	**0.380**	0.178	0.294	0.311

Improved by removing the unwanted factor. Table 3.6 and Table 3.7 show that: compared with standard clustering methods, our COUF achieves superior performance on all datasets. It verifies the claim that our COUF which explicitly removes the influence of the unwanted factor performs better than the standard clustering methods. Note that PICA obtains poor results since it conducts clustering on raw features (k-means on MoCo extracted feature achieves better results than PICA on raw features also reported in [Tsai et al., 2021]). And SPICE performs worse than IDEC because it applies a discriminative

[3]IDEC is a representative AE-based clustering method.

[4]PICA and SPICE are recently proposed self-supervised clustering methods. SPICE is the SOTA standard clustering method.

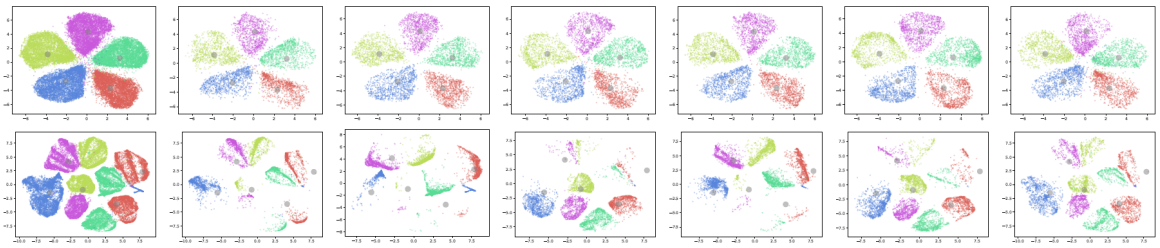


Figure 3.9 t-SNE on latent representations and clustering centroids from COUF (1st raw) and IDEC (2nd raw) on *Rotated Fashion*, respectively. The big grey dots are the centroids. The small dots are the representations, of which the colors denote the ground truth category labels.

model for clustering, which is more vulnerable to the unwanted factor than IDEC which is AE-based clustering.

To further illustrate the effectiveness of removing the unwanted factor, we visualize the latent representations and the clustering centroids for our COUF and IDEC (i.e., standard clustering that neglects the existence of the unwanted factor) on *Rotated Fashion*, respectively.

Invariant representations. From the t-SNE visualization of our COUF (the first row of Fig. 3.9), we can see that: (1) the clusters are well separated and the centroids are located at the center of each cluster. (2) These categories' representations are not only well aligned with each other, but also the whole data's representations. This demonstrates that our COUF's latent representations are invariant to the unwanted factor, i.e., the cloth category label. (3) Each centroid represents one of the five rotation angles in the dataset. In addition, the reconstruction of the centroids is exactly the Fashion-MNIST objects, which demonstrates our COUF captures semantic clustering structures.

The t-SNE visualization of IDEC (the second row of Fig. 3.9) shows that: (1) IDEC obtains an inferior clustering structure due to the negative impact of the unwanted factor. Specifically, the cloth category introduces variances into the data, making the derived structure away from the desired one w.r.t. the rotation factor. (2) These categories' representations are neither aligned with each other nor with the representation of the entire data. It demonstrates that IDEC's latent representations are corrupted by the unwanted factor, i.e., variances of cloth category.

Disentangled centroid reconstruction. We can reconstruct the centroids conditioned on the unwanted factor for COUF. The first row of Fig. 3.10 shows that (1) the latent embedding z and the unwanted factor c are well disentangled. In particular, the information of the unwanted factor is well captured by c. (2) The centroids from COUF can capture all rotation

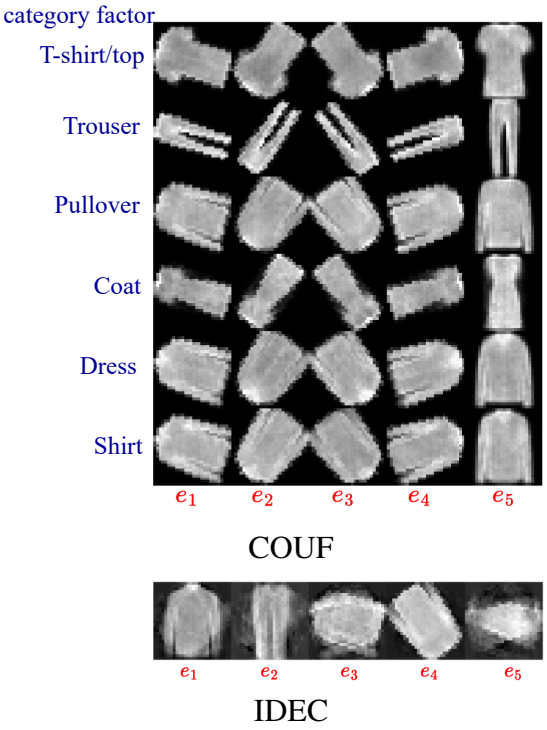


Figure 3.10 The centroids' reconstruction of COUF and IDEC on *Rotated Fashion* (28×28). Each column is conditioned on the same clustering centroid. Each row is conditioned on different labels of the cloth category factor.

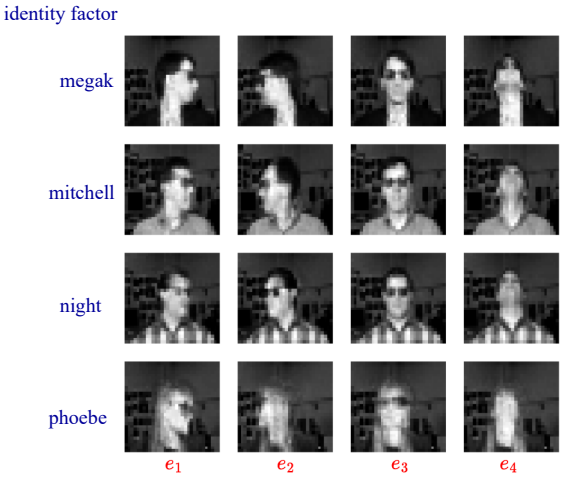


Figure 3.11 The centroids' reconstruction of COUF on *UCI-Face* (32×30). Each row is conditioned on different labels of the identity factor. Each column is conditioned on the same clustering centroid.

angles. When conditioned on the same clustering centroid, the reconstructions present the same angle.

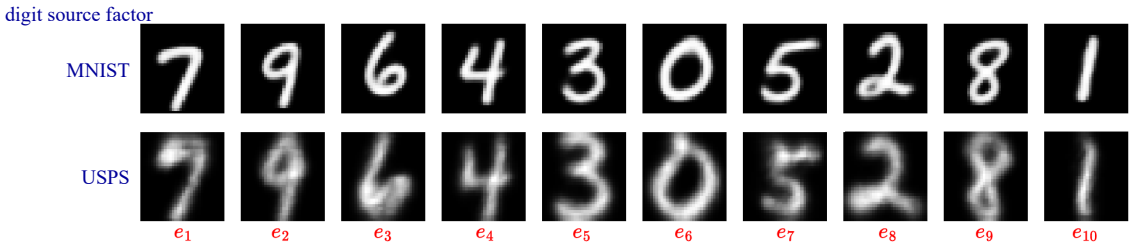


Figure 3.12 The centroids' reconstruction of COUF on *MNIST-USPS* (32×32). Each row is conditioned on different labels of the digit source factor. Each column is conditioned on the same clustering centroid.

The second row of Fig. 3.10 shows that (1) IDEC does not have the ability to disentangle the unwanted factor c from the latent space and cannot control the generation by being conditioned on c. (2) Its centroids do not capture all rotation angles in the dataset but some of them capture the cloth categories. For example, e_1 and e_2 represent the shirt and the trouser with the same angle, respectively.

We also reconstruct the centroids on *UCI-Face* and *MNIST-USPS* in Fig. 3.11 in Fig. 3.12, respectively. In their centroids' reconstruction, the latent embedding z and the unwanted factor c are also well disentangled. The centroids can capture clear structures, i.e., the pose angle for *UCI-Face* and the digit type for *MNIST-USPS*, respectively. On *Office-31* and *CIFAR10-C*, we do not reconstruct the centroids on these datasets as the extracted features are used as model input.

Ablation study

We study the effectiveness of each module by excluding it from our COUF framework.

Table 3.8 shows that: (1) our COUF gets the best results, which justifies the necessity of each module. (2) Without the disentanglement module to remove the unwanted factor via mutual information, the clustering performance drops significantly since the unwanted factor would distract desired clustering results. (3) A poor clustering structure is obtained without the clustering module because it fails to derive clustering-friendly representations. (4) The clustering performance is worse when excluding both the clustering module and the disentanglement module. (5) The other two modules cannot function without the VAE module. So we did not collect the result for COUF without the VAE module. Nevertheless, the VAE module allows clustering to be performed in the latent semantic space, which can have better clustering results[5].

[5]This can be said verified by the improvement of RUV_z compared to RUV_x in Section 3.2.2.

Table 3.8 Effectiveness of different modules in COUF on *Rotated Fashion*. "Clu" means the clustering module. "Dis" means the disentanglement module.

Metric	w/o Clu	w/o Dis	w/o VAE	w/o Clu & Dis	COUF
ACC	0.513	0.857	-	0.487	**0.985**
NMI	0.376	0.803	-	0.414	**0.940**

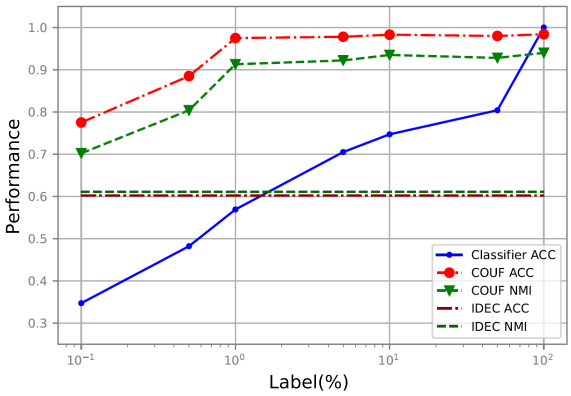


Figure 3.13 The clustering performance (ACC, NMI) of COUF given partial labels regarding the unwanted factor on *Rotated Fashion*. "Classifier ACC" denotes the test accuracy of the classifier. x axis denotes the proportion of labeled data. IDEC is the baseline that performs clustering without removing the unwanted factor.

Extension to the incomplete unwanted factor

We explore the performance of COUF given different amounts of labeled data w.r.t. the unwanted factor on *Rotated Fashion*. Applying COUF to this semi-supervised setting, we first train a classifier on the labeled data and use it to predict labels for the remaining unlabeled data. Then COUF is naturally applied to these fully-annotated data. Particularly, we employ a convolutional neural network (CNN) classifier for the classification. IDEC is adopted as the baseline following the same setting as COUF.

We plot the test accuracy of the classifier (calculated on the remaining unlabeled data) and the clustering performance (ACC and NMI) of COUF in Fig. 3.13 with the percentage of labeled data from 0.1% to 100%. It shows that (1) compared to IDEC which does not consider the unwanted factor, our COUF can improve the clustering performance even with a very small amount of labeled data. (2) When there are less than 0.5% labeled data, the test accuracy of the classifier is low, smaller than 0.5. Accordingly, the results of COUF are relatively not so good since there are more than 50% samples assigned with wrong labels. (3) When the labeled data is larger than 1%, there are more than 50% samples assigned with true labels. Though the percentage of label noise is still very high, COUF can perform well

since the correct labels dominate and the structured representations can be robust to label noise [Yu et al., 2020]. In conclusion, our COUF can work well even given a small amount of labeled data regarding the unwanted factor.

3.2.3 Summary

We have introduced a general framework COUF for a new stream of clustering that aims to deliver clustering results invariant to the pre-designated unwanted factor. COUF is the first deep clustering framework that eliminates unwanted factors in the semantic latent space of data through nonlinear dependency measures, which can more effectively remove unwanted information of complex data and thus obtain better clustering results. Our theoretical analyses reveal that COUF's losses are approximations to the mutual information between the representation and the unwanted factor as well as that between the representation and the interested clustering factor, which rigorously guarantees its efficacy in eliminating the unwanted factor and clustering. Empirical results demonstrate that COUF in general achieves better results on various image datasets than various baselines. In particular, COUF, i.e., a controllable DGM, consistently performs better than vanilla DGMs on all datasets for clustering tasks. In addition, COUF can generate desired samples specified by a certain factor or a clustering structure (Fig. 3.10, Fig. 3.11 and Fig. 3.12).

The future work can be explored from the following two directions. (1) We can extend our COUF to more situations as we make no assumptions specific to the form of data or the unwanted factor. Therefore, we can apply COUF to other types of data, e.g., text data, and time series data. In addition, COUF can be extended to use the unwanted factor with continuous values, such as the continuous "income" attribute in a human-related dataset [Redmond and Baveja, 2002]. (2) We can explore a joint training paradigm for the semi-supervised situation in which the unwanted factor is partially labeled, while we empirically studied the situation in a two-stage training paradigm (Section 3.2.2).

Chapter 4

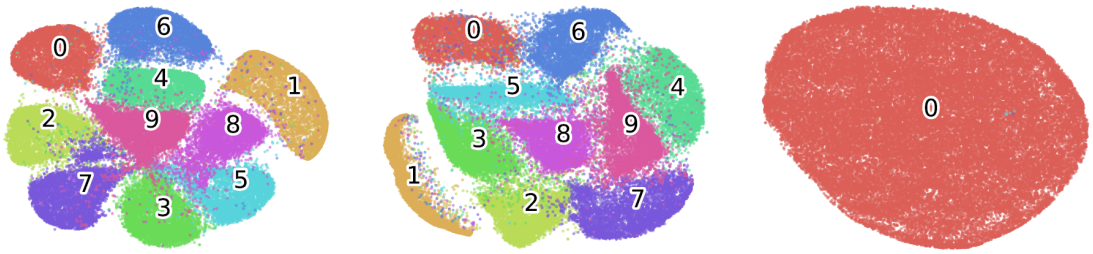
Incorporating Preferences into DGM

From this chapter, we start to delve into the main topic of this thesis (Fig. 1.2). We will study how human preferences can be incorporated into Wasserstein GAN (WGAN) in this chapter. As clarified in Chapter 3, WGAN would not suffer from the mode collapse issue, thus our framework built on it would maintain good sample diversity. Specifically, we propose Differential-Critic Generative Adversarial Network (DiCGAN) to learn the distribution of user-desired data when only partial instead of the entire dataset possesses the desired property. DiCGAN introduces a differential critic that learns from pairwise preferences, which are partial knowledge about the property and can be defined on a part of training data. The critic is built by defining an additional ranking loss over the WGAN's critic. It endows the difference of critic values between each pair of samples with the user preference and guides the generation of the desired data instead of the whole data. For a more efficient solution to ensure data quality, we further reformulate DiCGAN as a constrained optimization problem, based on which we theoretically prove the convergence of our DiCGAN. Extensive experiments on a diverse set of datasets with various applications demonstrate that our DiCGAN achieves state-of-the-art performance in learning the user-desired data distributions, especially in the cases of insufficient desired data and limited supervision.

4.1 Problem Statement

User-desired data may refer to some certain class of data among multiple class datasets, or observations with/without some particular attributes or properties. Such data can be induced from human preference, which can be represented as an ordering relation between two or more samples in terms of the desired property.

A universal criterion to help derive a user-desired data distribution can be constructed based on a score function. Following the score-based ranking literature [Cao et al., 2007a],



(a) Training data (b) GAN (c) DiCGAN

Figure 4.1 t-SNE of 50K MNIST samples from (a) training data, (b) GAN and (c) DiCGAN, respectively. Training on MNIST, DiCGAN learns the distribution of small digits, i.e., digit zero, while GAN learns the distribution of the entire dataset.

we suppose that there exists a numeric score associated with each sample, reflecting the user's preference for the sample. A higher score indicates that its corresponding sample is preferred by the user. In detail, let $f()$ denote a score function that maps sample x to score $f(x)$. Let T denote the threshold to discriminate the desired data from the undesired data. That is, if a sample's score $f(x)$ exceeds a predefined threshold T, namely, $I(f(x) > T) = 1$, the sample x is desired by the user. $I()$ is a sign function, which equals 1 if its condition is true and 0 otherwise. For the sake of explanation, we use $P_{\mathrm{r}}(x), P_{\mathrm{d}}(x), P_{\mathrm{u}}(x)$ to denote the distribution of the whole data, the user-desired data and the undesired data, respectively.

Current literatures [Gupta and Zou, 2019, Arjovsky et al., 2017b, Mirza and Osindero, 2014] needs to explicitly label desired/undesired data in order to learn the distribution of the desired data $P_{\mathrm{d}}(x)$. Namely, the desired data $\mathcal{X}_{\mathrm{d}} = \{x | I(f(x) > T) = 1, x \sim P_{\mathrm{r}}(x)\}$. The undesired data $\mathcal{X}_{\mathrm{u}} = \{x | I(f(x) \leq T) = 1, x \sim P_{\mathrm{r}}(x)\}$. However, the assumption that the score function $f()$ is predefined may be too restrictive for real applications, where no universal and explicit criteria exist. Second, the definitions of the desired/undesired samples are highly dependent on the choice of the threshold T. Third, labeling over the entire dataset incurs high costs.

Instead of relying on a predefined score function (global knowledge), we propose to learn the desired data distribution in a straightforward manner from the user preferences. Here, we consider general auxiliary information, i.e., the pairwise preferences, to represent the user preferences, due to its simplicity and easy accessibility. For any two samples $x_1, x_2 \sim P_{\mathrm{r}}(x)$, let $x_1 > x_2$ denote that x_1 is preferred over x_2 according to the user's preference over the samples. Let $\mathcal{X}$ be the training samples, i.e., $\mathcal{X} = \{x \sim P_{\mathrm{r}}(x)\}$. A collection of pairwise preferences $\mathcal{S}$ is obtained by:

$$\mathcal{S} = \{s = (x_1, x_2) | x_1 > x_2, x_1, x_2 \in \mathcal{X}\}. \tag{4.1}$$

S can be defined on part of the dataset.

Remark 1. *We can construct $\mathcal{S}$ by first randomly drawing sample pairs from part of the training samples and then asking the user to select the preferred one from each pair.*

Definition 2 (Problem Setting)**.** *Given the training samples $\mathcal{X}$ and the pairwise preferences $\mathcal{S}$, the target is to learn a generative model $P_{\mathrm{g}}(x)$ that is identical to the distribution of the desired data $P_{\mathrm{d}}(x)$, i.e., $P_{\mathrm{g}}(x) = P_{\mathrm{d}}(x)$ (See Fig. 4.1 for an example).*

4.2 Differential Critic GAN for User-Desired Distribution

Instead of adopting WGAN's critic for quality assessment, we present the differential critic for modeling pairwise preferences. The differential critic can guide the generation of the user-desired data.

4.2.1 Pairwise Preference

We consider incorporating pairwise preferences into the training of GAN.

The score-based ranking model [Zhou et al., 2008] is used to model the pairwise preferences. It learns the score function $f()$, of which the score value, called ranking score in the model, is the indicator of the user preferences. Further, the difference in ranking scores can indicate the pairwise preference relation. That is, for any pair of samples x_1, x_2, if $x_1 > x_2$ then $f(x_1) - f(x_2) > 0$ and vice versa. For any pairwise preference $s : x_1 > x_2$, the ranking loss we consider is as follows:

$$h(s) = \max\left(0, -\left(f\left(x_1\right) - f\left(x_2\right)\right) + m\right), \tag{4.2}$$

where m is the ranking margin. For other forms of ranking losses, the reader can refer to [Zhou et al., 2008].

Instead of learning the score function independently of GAN's training, we consider incorporating it into GAN's training, guiding GAN towards the generation of the desired data. The critic in RGAN [Jolicoeur-Martineau, 2019] is similar to the score function, where the critic values are used to describe the quality of samples. We are motivated to take the critic values as the ranking scores and define the ranking loss on the critic directly. In particular, the difference in the critic values for each pair of samples reflects the user's preference over the samples.

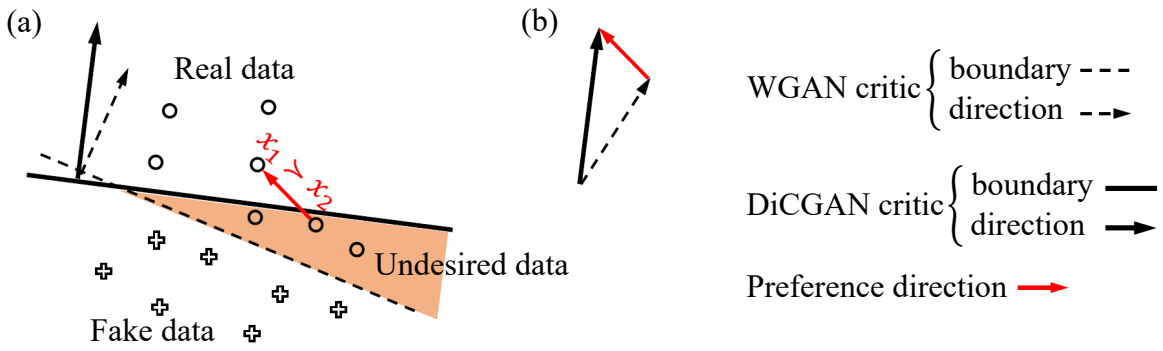


Figure 4.2 Illustration of why DiCGAN can learn the user-desired data distribution. (a) DiCGAN's critic pushes fake data towards the real desired data while WGAN's critic pushes fake data towards all the real data. (b) The change of DiCGAN's critic direction is driven by the preference direction. Note that the preference direction is learned from all pairwise preferences.

4.2.2 Loss Function

We build DiCGAN based on WGAN and the pairwise ranking loss is defined over the WGAN's critic. The loss function for DiCGAN is defined as:

$$\min_G \max_D \mathbb{E}_{P_r(x)}[D(x)] - \mathbb{E}_{P_g(x)}[D(x)] - \lambda \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} [h(s)], \tag{4.3}$$

where $h(s)$ is the pairwise ranking loss (Eq. (4.2)). $f()$ is approximated by the critic D. Namely, $h(s) \approx \max(0, -(D(x_1) - D(x_2)) + m)$. λ is a balance factor, which will be discussed further in section 4.3. Similar to WGAN, we formulate the objective for the differential critic L_D and the generator L_G as:

$$\begin{aligned} L_D &= \frac{1}{b} \sum_{i=1}^{b} (D(x_i) - D(G(z_i))) - \lambda \frac{1}{n_s} \sum_{j=1}^{n_s} h(s^j), \\ L_G &= \frac{1}{b} \sum_{i=1}^{b} -D(G(z_i)). \end{aligned} \tag{4.4}$$

where b is the batch size. n_s is the number of preferences sampling from $\mathcal{S}$.

The advantages of DiCGAN are twofold. (1) The introduced ranking loss in DiCGAN is defined on the critic directly. Apart from WGAN, it can be easily applied to other GAN variants developed based on the critic, e.g., RGAN. (2) The construction of pairwise preferences involves the undesired data. Thus, the undesired samples are also utilized during the training and they, together with desired samples, provide the generation direction of the desired data for the generator.

We argue that the differential critic in DiCGAN can guide the generator to learn the user-desired data distribution. As shown in Fig. 4.2, the differential critic in DiCGAN provides the direction towards the real desired data. We denote the critic direction as the moving direction of the fake data, which is orthogonal to the decision boundary of the critic. Referring to Eq. (4.3), DiCGAN's critic loss consists of two terms: the vanilla WGAN loss and the ranking loss. The vanilla WGAN loss imposes the critic direction from the fake data to the real data. Meanwhile, the ranking loss induces a user preference direction, which points from the undesired data to the desired data. Combining these two effects, the critic direction of DiCGAN targets the region of the real desired data only.

The above proposed DiCGAN (Eq. (4.3)) however requires sensitive hyperparameter tuning during the training. Revisiting the objective (Eq. (4.3)), the first two terms (WGAN loss) can be considered as the WGAN regularization, which ensures the generated data distribution is close to the whole real data distribution, i.e., $P_{\mathrm{g}} \approx P_{\mathrm{r}}$. The third term (ranking loss) serves as a correction for WGAN, which makes WGAN slightly biased to our target of learning the desired data distribution, i.e., $P_{\mathrm{g}} = P_{\mathrm{d}}$. Therefore, the WGAN regularization serves as the cornerstone of our DiCGAN. Particularly, if the desired data distribution is close to the whole data distribution, the ranking loss easily corrects the WGAN to achieve the desired data distribution. Otherwise, satisfactory performance of DiCGAN may require the online hyperparameter tuning of λ during the training process. Thus, it is hard to train with Eq. (4.3) in this case.

4.3 Reformulating DiCGAN to Ensure Data Quality

In this section, we reformulate DiCGAN as a form with a hard constraint. This form indicates that the tuning for Eq. (4.3) relies largely on the distance between the distributions of the desired data and the undesired data. Further, it inspires us to derive a more efficient solution – minor correction and major correction.

According to the above analysis, the WGAN loss serves as the cornerstone of our DiCGAN and the pairwise ranking loss serves as a correction for WGAN. Thus, we consider reformulating the objective of DiCGAN, i.e., Eq. (4.3) into an equivalent objective with a hard WGAN constraint:

$$\begin{aligned} &\min_{G}\max_{D} -\sum_{s\in\mathcal{S}}\left[h\left(s\right)\right], \\ &\text{s.t. } d(P_{\mathrm{r}},P_{\mathrm{g}}) = \left|\mathbb{E}_{P_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{P_{\mathrm{g}}(x)}\left[D\left(x\right)\right]\right| \leq \varepsilon. \end{aligned} \tag{4.5}$$

where $\varepsilon > 0$. Note that we impose an explicit non-negative constraint on $d(P_{\mathrm{r}}, P_{\mathrm{g}})$, to highlight that it is a distance metric. It is still equivalent to WGAN loss from its definition. Eq. (4.3)

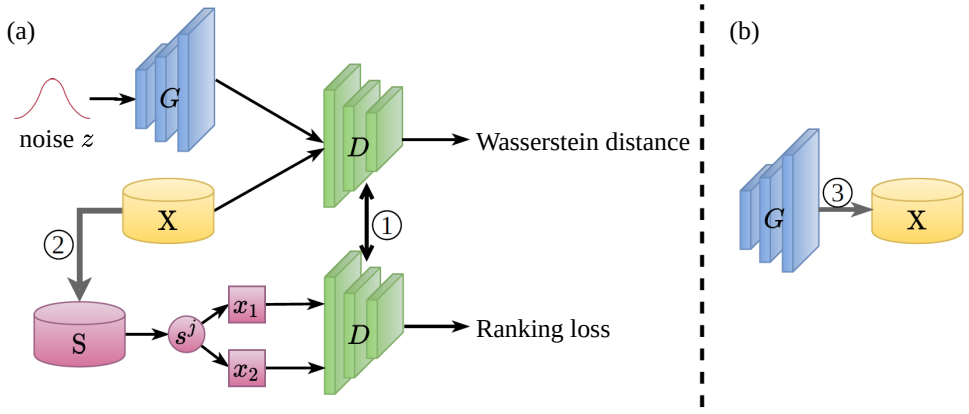


Figure 4.3 DiCGAN architecture and training. DiCGAN is alternately trained with step (a) and (b). (a) Training DiCGAN at one minor correction. (b) Replacing data after one minor correction. ① denotes the shared differential critic D. ② denotes that $\mathcal{S}$ is constructed from $\mathcal{X}$ using Eq. (4.1). ③ denotes data replacement using Eq. (4.7).

is the Lagrangian function. Since Eq. (4.5) imposes a hard constraint on the WGAN loss, it is more difficult to optimize compared to Eq. (4.3). However, more efficient solutions of DiCGAN can be explored by analyzing Eq. (4.5) regarding the hard constraint on $d(P_\mathrm{r}, P_\mathrm{g})$.

In terms of a minor correction situation, this means the desired data distribution P_d is close to the real data distribution P_r. Therefore, the hard constraint dominates the training goal of DiCGAN. By assigning a proper λ to ensure the constraint is satisfied, Eq. (4.3) can learn the distribution of the user-desired data while ensuring data quality.

In terms of a major correction situation, this means the desired data distribution P_d is quite diverse from the real data distribution P_r. Therefore, DiCGAN needs to achieve an equilibrium between the correction, imposed by the ranking loss, and the hard constraint, imposed by the WGAN loss. However, a large correction may not ensure the quality of the generated data, since the WGAN loss, used to guarantee the image quality, is defined between the generated data and the whole real data. To avoid the major correction, we propose to break the major correction into a sequence of minor corrections to ensure data quality. Namely, at each minor correction, we first use the generator G to generate n_g samples, denoted as $\mathrm{X_g}$:

$$\mathcal{X}_\mathrm{g}^e \leftarrow \{G^e(z^1), \ldots, G^e(z^{n_\mathrm{g}})\}, \quad \{z_i \sim p(z)\}_{i=1}^{n_\mathrm{g}}, \tag{4.6}$$

where e is e-th minor correction. Then we replace partial old training samples with the generated samples:

$$\mathcal{X}^{e+1} \leftarrow \mathcal{X}^e \setminus \mathcal{X}_\mathrm{o}^e \cup \mathcal{X}_\mathrm{g}^e, \tag{4.7}$$

where $\mathcal{X}_\mathrm{o}^e$ are the old (least-recently added) n_g samples in $\mathcal{X}^e$.

Algorithm 2 Training algorithm of DiCGAN

Input: training data $\mathcal{X}$, pairwise preferences $\mathcal{S}$
Initilization: balance factor λ, #generated samples n_{g}, #pairs n_{s}, batch size b, #iterations per minor correction n_{minor}, #critic iterations per generator iteration n_{critic}
Pretrain D and G
repeat
  % Shift to the user-preferred distribution
    Generate samples using Eq. (4.6)
    Replace partial old samples in $\mathcal{X}$ with $\mathrm{X_g}$ using Eq. (4.7)
  Obtain pairwise preferences $\mathcal{S}$ using Eq. (4.1)
  % Training of D and G at a minor correction
  for $l = 1, \ldots, n_{\mathrm{minor}}$ **do**
    for $t = 1, \ldots, n_{\mathrm{critic}}$ **do**
      Sample $\{x_i\}_{i=1}^{b}$ from $\mathcal{X}$, $\{z_i \sim p(z)\}_{i=1}^{b}$
      Sample $\{s^j\}_{j=1}^{n_{\mathrm{s}}}$ from $\mathcal{S}$.
      Train the differential critic D using L_D in Eq. (4.4)
    end for
    Train the generator G using L_G in Eq. (4.4)
  end for
until converge
Output: generator G for desired data distribution

Due to the ranking loss, the generated data distribution P_{g}^{e} is closer to the desired data distribution P_{d}, compared to the constructed P_{r}^{e} at each minor correction. Therefore, the iterative replacement (Eq. (4.7)) can gradually shift the real data distribution P_{r} towards the desired data distribution P_{d}. Namely, $d(P_{\mathrm{r}}, P_{\mathrm{d}}) > \cdots > d(P_{\mathrm{r}}^{e}, P_{\mathrm{d}}) > d(P_{\mathrm{r}}^{e+1}, P_{\mathrm{d}}) > \cdots$. According to the monotone convergence theorem, $d(P_{\mathrm{r}}^{e}, P_{\mathrm{d}})$ will converge to zero when $e \to +\infty$. So only a minor correction needs to be imposed on P_{g}^{e} by optimizing Eq. (4.3) at each minor correction. Iteratively, the generated distribution P_{g} shifts towards P_{d}. The training algorithm is summarized in Algorithm 2. The architecture and training of DiCGAN can be seen in Fig. 4.3. For the sake of easy optimization, we pretrain the differential critic D and the generator G using vanilla WGAN.

4.4 Convergence Analysis

In this section, we analyze the convergence of our DiCGAN under the minor correction and the major correction, respectively. In the case of the minor correction, we prove that the distribution of generated data $P_{\mathrm{g}}(x)$ converges to the distribution of user-desired data $P_{\mathrm{d}}(x)$ via adversarial training of GAN given that the differential critic of DiCGAN converges to the score function whose score describes the user's preference for the sample. In the case of the

major correction, $P_g(x)$ is proven gradually moving towards $P_d(x)$ with a sequence of minor correction as one minor correction shifts $P_g(x)$ towards $P_d(x)$ with a certain small distance.

Suppose the training data $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, where n is the number of training samples. Their corresponding scores $o = f(x)$ are $\{o_1, o_2, \ldots, o_n\}$, which describes the user's preference for the sample. The maximum score among the samples is denoted as o_{max} while the minimum one is o_{min}.

Proposition 2. *In the case of the minor correction, i.e., $d(P_r(x), P_d(x)) \leq \varepsilon$, $P_g(x)$ converges to $P_d(x)$.*

Proof. According to the theory of learning to rank [Liu, 2009], by setting an appropriate λ, we have D converge to the score function $f()$ iff S is sufficient. Then the real desired data will be assigned higher scores than the real undesired data.

With Eq. (4.4), the generator is optimized to generate samples with scores as high as possible while the critic is optimized to assign the generated samples lower than the training samples. As only training samples preferred by the user are assigned with high scores, when the adversarial training converges, the generated samples are alike samples with high scores, i.e., the desired data, which shares the same principle in [Goodfellow et al., 2014, Arjovsky et al., 2017b]. Therefore, $P_g(x)$ converges to $P_d(x)$. □

Proposition 3. *In the case of the minor correction where $d(P_r(x), P_d(x)) \leq \varepsilon$ and $P_g(x) = P_d(x)$, we can prove that $\mathbb{E}_{P_g(x)}[D(x)] = \mathbb{E}_{P_r(x)}[D(x)] + \delta$, for some $\delta > 0$.*

The proof is left in Appendix B.

Corollary 2. *The minor correction moves P_g towards P_d with distance δ compared to P_r, i.e., $d(P_r, P_d) - d(P_g, P_d) = \delta$.*

Proposition 4. *In the case of the major correction, i.e., $d(P_r, P_d) = T_0$, the distance between $P_g(x)$ and $P_d(x)$ converges to $d(P_g^k, P_d) = T_0 - k\delta$ after k minor corrections.*

The proof is put in Appendix B

Corollary 3. *In the case of major correction, $P_g(x)$ converges to $P_d(x)$ after K minor corrections, where $K = \lceil \frac{T_0}{\delta} \rceil$.*

Proof. Since $d(P_g^k, P_d) >= 0$ and $d(P_g^k, P_d) = T_0 - k\delta$ decreases as k increases, $d(P_g^k, P_d)$ converges to zero when $k \to +\infty$ according to the monotone convergence theorem. Specifically, when $k = \lceil \frac{T_0}{\delta} \rceil$, $d(P_g^k, P_d) = 0$. □

From Proposition 2 and Corollary 3, we conclude that in DiCGAN, the distribution of generated samples $P_g(x)$ converges to $P_d(x)$.

4.5 Discussions

In this section, we discuss the technical novelty and some possible variants of DiCGAN.

4.5.1 Technical Novelty of DiCGAN

Our DiCGAN's technical novelty and its significance in terms of the following four aspects:

- **The first one to apply user preferences for desired data generation.** Current approaches for desired data generation require expensive global knowledge, which is usually not available. Our DiCGAN uses local knowledge only – local ranking information about user preferences.
- **New insight for critic value.** Our DiCGAN considers the critic values as the ranking scores that represent user preferences. Based on this insight, we can incorporate user preferences into GAN's learning instead of learning the score function for user preferences independently of GAN's training:
  1. *Naive combination between user preferences and GAN does not work.* Introducing an additional critic that learns from user preferences onto WGAN would lead to the conflict between WGAN's original critic for good quality generation and the extra critic for desired data generation.
  2. As critic values can represent data quality and user preferences, we define a differential critic by defining an additional pairwise ranking loss on the WGAN's critic and build DiCGAN (Eq. (4.3)). Then the original WGAN's critic loss encourages:

     $$x_1 > x_2 \text{ for } x_1 \sim P_{\mathrm{d}}(x) \text{ and } x_2 \sim P_{\mathrm{u}}(x);$$

     and the ranking loss encourages:

     $$x_1 > x_2 \text{ for } x_1 \sim P_{\mathrm{r}}(x) \text{ and } x_2 \sim P_{\mathrm{g}}(x).$$

     The critic would guide the generation with high critic values, encouraging the generation of user-desired data with good quality.
- **Efficient solution by an equivalent form with a hard constraint.** The naive form of DiCGAN (Eq. (4.3)) requires heavy hyper-parameter tuning when there is a large distance gap between the distributions of the desired data and the whole data. Thus, we propose an equivalent form of DiCGAN (Eq. (4.5)). Based on it, we derive a more efficient solution in terms of minor correction and major correction, which can always ensure good data quality.

- **The first rigorous model for desired data generation.** To the best of our knowledge, no previous work theoretically studies this problem. The above three points pave the way for the theoretical convergence proof of desired data generation:

1. Because of DiCGAN's form with a hard distance constraint, we can analyze the convergence of DiCGAN under the minor correction and the major correction.
2. Since we interpret the critic values in DiCGAN as the ranking scores, the relationship between the user preferences (reflected by ranking scores) and the distribution distance (represented by critic values) [Proposition 3 and Corollary 2] can be derived. This is the first time that such a relationship is rigorously shown.

4.5.2 Pairwise Regularization to Generator

We claim that adding the pairwise regularization to the generator requires heavy supervision and is invalid. Our DiCGAN thus does not consider such regularization.

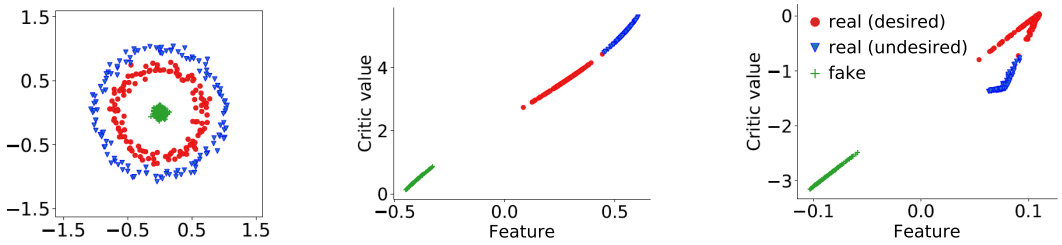
As the target is to learn the desired data distribution, the regularization on the generator can be used to make the critic values of the generated samples larger than those of the undesired samples. Specifically, a selector is first applied to give a full ranking for the training data, and then the bottom K_0 samples are picked up as the undesired samples. The pairwise preferences are then defined over the generated samples and the undesired samples. Note that the undesired subset of the training data requires labeling all training data.

We consider two cases of adding the regularization to the generator. First, we only add the pairwise regularization to the generator (PRG-1). Second, we add the regularization to the generator together with the regularization on the critic (PRG-2).

The objective for PRG-1 is as follows:

$$\begin{aligned} L_D &= \mathbb{E}_{P_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{P_{\mathrm{g}}(x)}\left[D\left(x\right)\right], \\ L_G &= \mathbb{E}_{P_{\mathrm{g}}(x)}\left[D\left(x\right)\right] - \lambda' \frac{1}{|\mathcal{S}'|} \sum_{s \in \mathcal{S}'} \left[h\left(s\right)\right], \end{aligned} \tag{4.8}$$

where $h(s)$ is Eq. (4.2). $\mathcal{S}'$ is the pairwise preferences constructed between the generated data and the undesired data, i.e., $\mathcal{S}' = \left\{s = (x_1, x_2) | x_1 > x_2, x_1 \sim P_{\mathrm{g}}(x), x_2 \sim P_{\mathrm{u}}(x)\right\}$. Now the generator consists of two terms, the original WGAN loss on the generator aims to achieve $\mathbb{E}_{P_{\mathrm{g}}(x)}\left[D\left(x\right)\right] > \mathbb{E}_{P_{\mathrm{r}}(x)}\left[D(x)\right]$, while the regularization aims to achieve $\mathbb{E}_{P_{\mathrm{g}}(x)}\left[D\left(x\right)\right] > \mathbb{E}_{P_{\mathrm{u}}(x)}\left[D(x)\right]$. Since the undesired data is a subset of the real data, i.e., $\{x | x \sim P_{\mathrm{u}}(x)\} \subseteq \{x | x \sim P_{\mathrm{r}}(x)\}$, the WGAN loss always dominates the training of the generator. Therefore, PRG-1 degenerates to WGAN.



(a) Data (b) WGAN (c) DiCGAN

Figure 4.4 Comparison of the critic in (b) WGAN and (c) DiCGAN. DiCGAN's critic can assign higher critic values for real desired data than real undesired data while WGAN's critic cannot. "Feature" is obtained by using kernel PCA to project the output on the second last layer of the critic into 1D space.

The objective for PRG-2 is as follows:

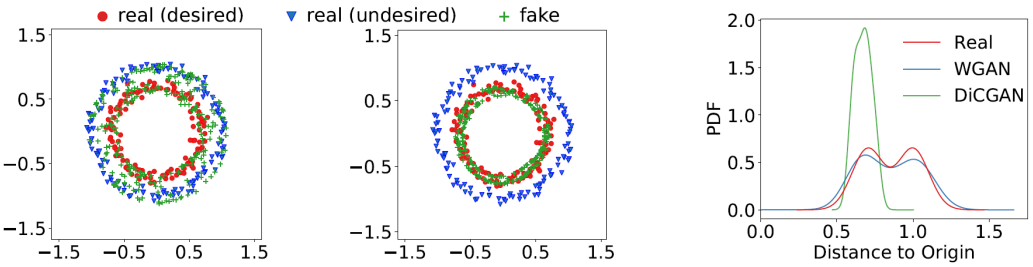
$$L_D = \mathbb{E}_{P_r(x)}\left[D(x)\right] - \mathbb{E}_{P_g(x)}\left[D\left(x\right)\right] - \lambda \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left[h\left(s\right)\right],$$
$$L_G = \mathbb{E}_{P_g(x)}\left[D\left(x\right)\right] - lambda' \frac{1}{|\mathcal{S}'|} \sum_{s \in \mathcal{S}'} \left[h\left(s\right)\right], \tag{4.9}$$

where $\mathcal{S}$ is constructed based on (4.1). Although the generator consists of two terms, the same as our analysis about PRG-1, the extra pairwise regularization on the generator is invalid. Meanwhile, the extra pairwise regularization on the critic works like that in DiCGAN. Therefore, the whole framework degenerates to DiCGAN.

4.6 Case Study on Synthetic Data

To gain an intuitive understanding of the differences between our DiCGAN and WGAN regarding the critic and the generator, we conduct a case study on a synthetic dataset.

The synthetic dataset consists of two concentric circles by adding Gaussian noise with a standard deviation of 0.05, which is a 2D mixture Gaussian distribution with two modes (See Fig. 4.4a). The samples located on the inner circle are considered to be the desired data, while the samples on the outer circle are defined as the undesired data. By labeling the desired data as $y = 1$ and the undesired data as $y = 0$, we can construct the pairwise preference for two samples x_1 and x_2 based on their labels. Namely, $x_1 \succ x_2$ if $y_1 = 1 \wedge y_2 = 0$, and vice versa. The pairs are constructed within each mini-batch. Our target is to learn the distribution of the desired data (i.e., samples on the inner circle), using the whole data along with the constructed pairwise preferences.



(a) WGAN (b) DiCGAN (c) PDF vs. distance

Figure 4.5 (a-b) Visualization of the generated samples from WGAN and DiCGAN. The fake data is expected to overlap with the real desired data only. (c) Probability density function (PDF) vs. sample distance to the origin.

4.6.1 WGAN vs.DiCGAN on Critic

Experiment setting: we fix the generator and simulate the fake data as the 2D Gaussian blob with a standard deviation of 0.05 (green pluses). We first train the critic until convergence. Then, we project the output on the second last layer of the critic into 1D space using kernel principal components analysis (PCA), to obtain the projected features. To explore the difference between the critics of WGAN and DiCGAN, we draw the curve of the critic values versus the projected features for WGAN and DiCGAN, respectively (Fig. 4.4b, 4.4c).

From Fig. 4.4b, 4.4c, we can see: (1) in terms of the real data and the fake data, the critic of both WGAN and DiCGAN can achieve perfect discrimination. Meanwhile, the projected features of the real data and those of the fake data are also completely separated; (2) in terms of the real desired data and the real undesired data, the critic of DiCGAN assigns higher values to the desired samples, compared to the undesired samples. This is because our ranking loss expects a higher ranking score (i.e., critic value) for the desired sample. (3) In contrast, the critic of WGAN assigns lower values to the desired data since the desired data is closer to the fake data compared to the undesired data.

4.6.2 WGAN vs. DiCGAN on Generator

Experiment setting: we train the critic and the generator following the regular GANs' training procedure. The generation results of WGAN and DiCGAN are shown in Fig. 4.5a, 4.5b.

DiCGAN (shown in Fig. 4.5b) only generates the user-desired data. Namely, generated data covers the inner circle. In contrast, WGAN (shown in Fig. 4.5a) generates all data. Namely, generated data covers the inner circle and the outer circle. As the critic in DiCGAN can guide the fake data towards the real data region and away from the undesired data region, the generator thus produces data that is similar to the real desired data. Be-

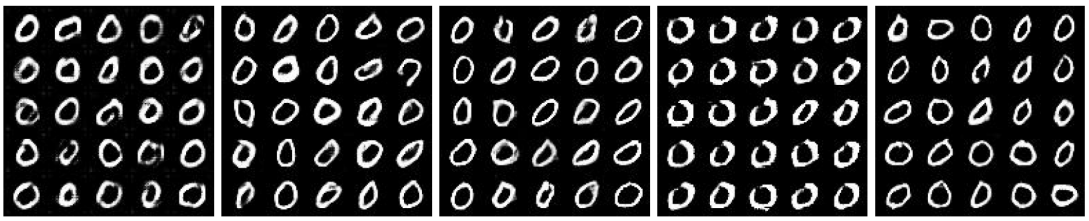
cause the critic in WGAN pushes the fake data to the region of all real data, the generator finally produces the whole real-alike data.

Further, we calculate the distance from the real samples to the origin and plot the probability density function versus the distance in Fig. 4.5c. We also do this for the generated samples from WGAN and DiCGAN, respectively. It shows that DiCGAN only captures one mode of the real data distribution, consistent with the results that DiCGAN only produces desired samples. In contrast, WGAN captures all modes of the real data distribution, meaning that WGAN generates all real data.

4.7 Experimental Study

Our DiCGAN for desired data generation has various applications in the real world. In particular, we apply our DiCGAN to two applications: 1) generating images that meet the user's interest for a given dataset, which can be used for image search [Yu and Kovashka, 2020]. 2) optimizing biological products with desired properties, which can automate the process of designing DNA sequences for usage in medicine and manufacturing [Gupta and Zou, 2019]. In these applications, we verify that our DiCGAN only using local knowledge (i.e., user preferences) outperforms current methods relying on global knowledge when labels of desired data are limited. Furthermore, we study the relation between critic values and user preferences as well as the effects of each component in DiCGAN.

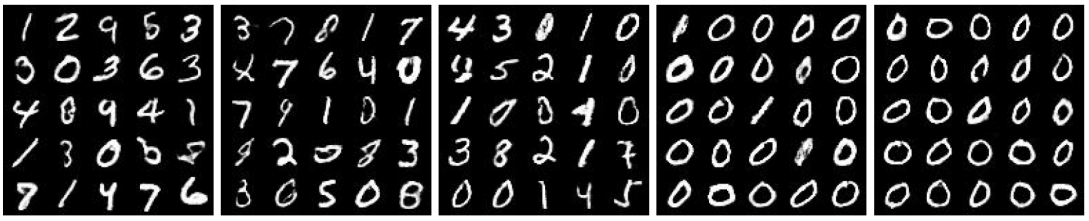
Baselines We compare DiCGAN with WGAN [Arjovsky et al., 2017b], CWGAN [Mirza and Osindero, 2014], FBGAN [Gupta and Zou, 2019] and GAN-FT. 1) WGAN is trained with only the desired data to derive the desired data distribution. 2) CWGAN is the extension of GAN with a conditional label c. To train CWGAN, we split the training data into the desired class ($c = 1$) and the undesired class ($c = 0$) based on global knowledge. Then $p(x|c = 1)$ is the desired data distribution. 3) FBGAN adopts an iterative training paradigm to derive the desired data distribution. First, FBGAN is pre-trained with all training data. At each training epoch, FBGAN resorts to an extra selector to select the desired samples from the generated samples and use them to replace the least-recently added samples in the training dataset. Then FBGAN performs regular GAN training with the updated training data. 4) GAN-FT is to fine-tune a pre-trained GAN with a classification loss on desired data. It is possible to use GAN loss defined between the generated data and the desired data to constrain the quality of desired data during the fine-tuning of GAN-FT. This is actually similar to the baseline WGAN that is trained on the desired subset of training data. Thus it would still suffer from poor data quality issues when there is limited desired data in the training dataset.



(a) WGAN (b) CWGAN (c) FBGAN (d) GAN-FT (e) DiCGAN

Figure 4.6 Generated images on MNIST by (a) WGAN, (b) CWGAN, (c) FBGAN, (d) GAN-FT and (e) DiCGAN

.



Iter 0 (1.0%) Iter 200 (16.1%) Iter 400 (28.2%) Iter 1000 (95.8%) Iter 2000 (99.9%)

Figure 4.7 Generated images of DiCGAN on MNIST during the training process. DiCGAN learns the distribution of small digits, which gradually generates more small digit images. The % denotes the percentage of zero digits in $50K$ generated samples.

Datasets MNIST [Lecun et al., 1998] consists of 28×28 images with digit zero to nine. $50K$ training images are regarded as training data. CelebA-HQ [Karras et al., 2018] is the high-quality subset of Celeb Faces Attributes Dataset, which has $30K$ face images of celebrities. We use all images as the training data and resize them to 64×64. The gene sequence dataset [Gupta and Zou, 2019] contains $3,655$ gene sequences with a maximum length of 156 codings for proteins collected from the Uniprot database. All methods applied to the datasets use the same supervision for a fair comparison. On MNIST and CelebA-HQ, we resort to class labels to derive the desired data distribution. On the gene sequence dataset, we resort to an analyzer that can evaluate the desired property for genes to derive the desired data distribution.

Remark 2. *Considering pairwise preferences over explicitly labeling what the user considers to be good data or not is beneficial especially given the limited supervision, which will be verified in the following experiments.*

Evaluation Metric: To evaluate the performance of learning the desired data distribution, we calculate the percentage of desired data (PDD) in GAN's generation. PDD $= \frac{|\{x|x \text{ is desired}, x \in \mathrm{X_g}\}|}{|\mathrm{X_g}|} \times 100\%$, where $\mathrm{X_g}$ are generated samples.

4.7.1 Capturing Small Digits on MNIST

Suppose the user is interested in learning the distribution of small digits on MNIST. Zero is the smallest digit of MNIST, thus as the desired data.

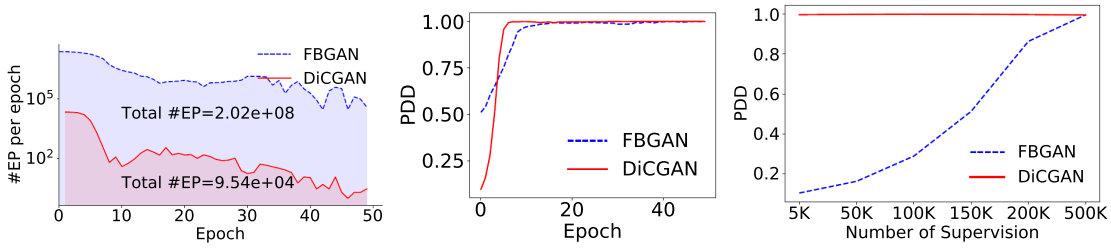
Networks & Hyperparameters By a coarse grid search, the balance factor λ is set to 1. The ranking margin m is set to 1 following [Cao et al., 2006]. The batch size b is set to 50. The network architecture of the critic and generator in our DiCGAN are based on WGAN-GP [Gulrajani et al., 2017]. See Supplementary for details. The baselines share the same architecture for a fair comparison. The optimizer is Adam [Kingma and Ba, 2015] with a learning rate of $1e$-4 and $\beta_1 = 0.5, \beta_2 = 0.9$. The number of critic iterations per generator iteration n_{critic} is 5.

Training As for WGAN and CWGAN, zero digits in the training data are regarded as the desired samples ($c = 1$), whose size is $4,950$. The other digits are labeled as the undesired samples, whose size is $45,050$ ($c = 0$). WGAN is only trained with the desired data. CWGAN conditions on c to model a conditional data distribution $p(x|c)$ for MNIST. For GAN-FT, we first pre-trained WGAN-GP with all digit images. Then we fine-tuned its generator with a classifier loss that makes the generated samples classified as digit zero. FBGAN and our DiCGAN both introduce the generated samples into the training dataset during the training. The labels of the generated samples are obtained by resorting to a classifier, pre-trained for digit classification. At every training epoch, FBGAN generates $50K$ samples and requests the classifier to label them. Then the selector in FBGAN will rank the images using their corresponding labels, where the smaller digits are ranked higher. The selector selects the generated images with digits ranked in the top 50%, i.e., small digits, as the desired data to replace old training data. As for DiCGAN, the pairwise comparison can be obtained for two images x_1 and x_2 according to their predicated label y_1 and y_2, namely $x_1 > x_2$ if $y_1 < y_2$, and vice versa. At each iteration, #pairwise preferences n_s is 25. #iteration per minor correction $n_{\text{minor}} = 200$. #generated samples for each minor correction $n_{\text{g}} = 50K$.

Fig. 4.7 presents the generated MNIST images randomly sampled from the generator of DiCGAN. It shows that the generated MNIST digits gradually shift to smaller digits during the training, and converge to the digit zero. For each method, we sample $50K$ samples from the generator and calculate the percentage of digit zero and digits zero to four among the

Table 4.1 Percentage of desired data in the generation (PDD) of various GANs on MNIST. Best results are highlighted in bold. Top 1 means digit zero. Top 5 means digits zero to four.

Method	Original	WGAN	CWGAN	FBGAN	GAN-FT	DiCGAN
Top 1	9.9	97.3	95.0	**100.0**	**100.0**	**100.0**
Top 5	51.1	98.2	96.4	**100.0**	**100.0**	**100.0**



(a) (b) (c)

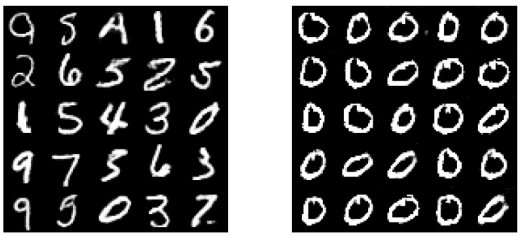
Figure 4.8 Comparison of DiCGAN and FBGAN on MNIST. (a) plots used #EP per epoch. (b) plots PDD versus the training epoch. (c) plots PDD versus the number of supervision.

generated digits for quantitative evaluation. In Table 4.1, only small digits are generated by DiCGAN and FBGAN; WGAN and CWGAN can also learn the distribution of the desired digit since the dataset is simple and has relatively sufficient data for the desired digit. The visual results shown in Fig. 4.6 are consistent with the quantitative results. However, when the dataset is complex and the desired data is insufficient, WGAN and CWGAN fail, which is described in Sect. 4.7.2. GAN-FT also only generates digit zero, but it suffers from mode collapse problem. The generated images have low diversity (Fig. 4.6d). This is because there lacks data quality guarantee during the later fine-tuning stage.

Comparison of DiCGAN and FBGAN

Though FBGAN achieves good performance in learning the desired data distribution, it requires a lot of supervision information from the selector. We calculate the number of effective pairs (#EP) used in DiCGAN and FBGAN, respectively. #EP in DiCGAN denotes the total number of explicitly constructed pairs during the training, i.e., $\#\text{EP} = \sum_{i=1}^{n_\text{e}} \sum_{j=1}^{n_\text{minor}} n_\text{s}$. As for FBGAN, its selector ranks all generated samples and selects the desired samples from them at each epoch. Therefore, #EP can be induced by the implicit pairs implied by the desired generated samples versus the undesired generated samples, i.e., $\#\text{EP} = \sum_{i=1}^{n_\text{e}} n_\text{gd} \times n_\text{gu}$, where n_e is the number of training epochs. where n_gd and n_gu denote the number of desired samples and undesired samples in the generation, respectively.

Fig. 4.8a plots FBGAN's and DiCGAN's used #EP at each epoch, respectively. It shows that (1) the #EP used in DiCGAN is much smaller than that in FBGAN at each training epoch; (2) the total #EP used in DiCGAN is significantly less than that in FBGAN, which can be reflected from the shadow area. In total, DiCGAN used $9.53e4$ effective pairs while FBGAN used $2.02e8$ effective pairs. Our DiCGAN is scalable to the large training dataset, e.g. MNIST. #EP in DiCGAN is linearly correlated to the training size. In contrast, #EP in FBGAN is determined by n_gd and n_gu, which are both linearly correlated to the training size. Thus, #EP in FBGAN is quadratically correlated to the training size.



(a) FBGAN (b) DiCGAN

Figure 4.9 The generated results of (a) FBGAN and (b) DiCGAN on MNIST given limited supervision.

We plot the ratio of digit zero in the generated data (PDD) of DiCGAN and FBGAN during the training process in Fig. 4.8b. It shows that DiCGAN converges faster than FBGAN.

Comparing DiCGAN and FBGAN given the limited supervision

We conduct the experiment on MNIST. Specifically, the query amount of resorting to the pre-trained classifier to obtain the prediction of the generated samples is restricted to $5K$ for both FBGAN and DiCGAN.

Table 4.2 shows that DiCGAN can learn the desired data distribution, generating 99.7% zero digits, while FBGAN fails, generating 10.3% digit zero, which is consistent with the visual results in Fig. 4.9a and Fig. 4.9b.

Table 4.2 PDD on MNIST given limited supervision.

Method	Top 1	Top 5
FBGAN	10.3	52.6
DiCGAN	**99.7**	**99.9**

We explore the gap in the performance between DiCGAN and FBGAN evolves as the number of supervision increases on MNIST. Specifically,the query amount of resorting to the pre-trained classifier to obtain the prediction of the generated samples is restricted to $5K, 50K, 100K, 150K, 200K, 500K$ for both FBGAN and DiCGAN.

Fig. 4.8c plots PDD versus the number of supervision for FBGAN and DiCGAN, respectively. It shows that (1) DiCGAN always learns the desired data distribution even given the limited supervision; (2) when given the limited supervision, FBGAN fails to learn the desired data distribution, i.e., achieving a small PDD; (3) FBGAN performs better and achieves a higher PDD, narrowing the performance gap with DiCGAN as the number of supervision increases.

Table 4.3 Percentage of desired data in the generation (PDD) and image quality (FID) of various GANs on CelebA-HQ. The best results are highlighted in bold. The second best results are underlined. The strikethrough on PDD of WGAN and GAN-FT denotes that they suffer from severe low-quality issues (large FID), generating very blur face images (Fig. 4.10) and thus its PDD is not really meaningful.

Method	Original	WGAN	CWGAN	FBGAN	GAN-FT	DiCGAN	DiCGAN$_{\text{style}}$
PDD	22.1	~~**76.0**~~	8.3	24.7	~~**99.7**~~	<u>33.4</u>	**57.0**
FID	-	115.4	79.7	51.6	107.1	<u>49.7</u>	**36.5**

4.7.2 Capturing Old Face Images on CelebA-HQ

Suppose the user is interested in learning the distribution of old face images on CelebA-HQ.

Networks & Hyperparameters The balance factor λ and the ranking margin m is set to 1. The batch size b is set to 64. The network architecture of the critic and generator in our DiCGAN are based on WGAN-GP [Gulrajani et al., 2017]. See Appendix for details. The baselines share the same architecture for a fair comparison. The optimizer is Adam with a learning rate of $2e$-4 and $\beta_1 = 0.5, \beta_2 = 0.999$. n_{critic} is set to 5. Further, we use an advanced GAN architecture, StyleGAN (*https://github.com/NVlabs/stylegan2*) [Karras et al., 2020b] to implement our DiCGAN, denoted as DiCGAN$_{\text{style}}$. The networks are optimized with Adam with $\beta_1 = 0, \beta_2 = 0.9$. The generator G's learning rate is $1e$-4 while the critic D's is $3e$-4 [Heusel et al., 2017]. n_{critic} is set to 1.

Training There are $6,632$ old face images, labeled as desired, and $23,368$ young face images, labeled as undesired, in the training data. WGAN is only trained with the constructed desired dataset. CWGAN conditions on c to model a conditional data distribution $p(x|c)$. A classifier, pre-trained for classifying young faces and old faces, is adopted for predicting the labels for the generated face images. Particularly, the query amount of resorting to the classifier is restricted to $30K$. As for FBGAN, at every training epoch, FBGAN generates $5K$ images, and those classified as old faces are selected by the selector to replace the old training data. As for GAN-FT, we first pre-trained WGAN-GP with all images including young faces and old faces. Then we fine-tuned the pre-trained generator with the classifier loss that makes the generated samples classified as old faces. As for DiCGAN, the generated face image classified as an old face is preferred over the face image classified with the young attribute. At each iteration, n_s is set to 64. n_{minor} is set to $1K$. n_{g} is set to $1K$. As for DiCGAN$_{\text{style}}$, at each iteration, n_s is set to 64. n_{minor} is set to 500. n_{g} is set to $30K$.

We visualize the generated face images randomly sampling from the generator of each model in Fig. 4.10. For each model, we sample $50K$ samples from the generator and then calculate the percentage of old face images (PDD) and the image quality score, i.e.,

(a) WGAN (6/9) (b) CWGAN (1/9) (c) FBGAN (3/9)

(d) GAN-FT (9/9) (e) DiCGAN (4/9) (f) DiCGAN$_{\text{style}}$ (6/9)

Figure 4.10 Generated images on CelebA-HQ by (a) WGAN, (b) CWGAN, (c) FBGAN, (d) GAN-FT, (e) DiCGAN and (e) DiCGAN$_{\text{style}}$. The red boxes refer to the images which are classified as old images.

Frechet Inception Distance (FID) among the generated samples for quantitative evaluation in Table 4.3. From Fig. 4.10 and Table 4.3, (1) though WGAN mainly generates desired data, it has poor generation since its training data only consists of the desired subset, and thus is insufficient, which has only $6,632$ face images. The generated face images are blurred. Meanwhile, WGAN's FID score is the highest among all methods, i.e., 115.4, quantitatively showing the poorest generation quality. (2) CWGAN has better generation quality than WGAN as it is trained with sufficient training data, $30K$ samples, but fails to shift towards the desired data distribution. There is only one old face image out of 9 randomly sampled images in the visualization result. Its PDD (8.3%) is smaller than the training data (Original, 22.1%). This is because the undesired data, i.e, the majority in the training data, dominates the generation of CWGAN. (3) FBGAN can achieve relatively good quality, with relatively small FID, but only slightly shift towards the distribution of desired data (PDD=24.7%) due to limited supervision. (4) GAN-FT almost generates old faces, but the quality is poor, verified by a large FID quality score and low-quality visual results in Fig. 4.10d. (5)

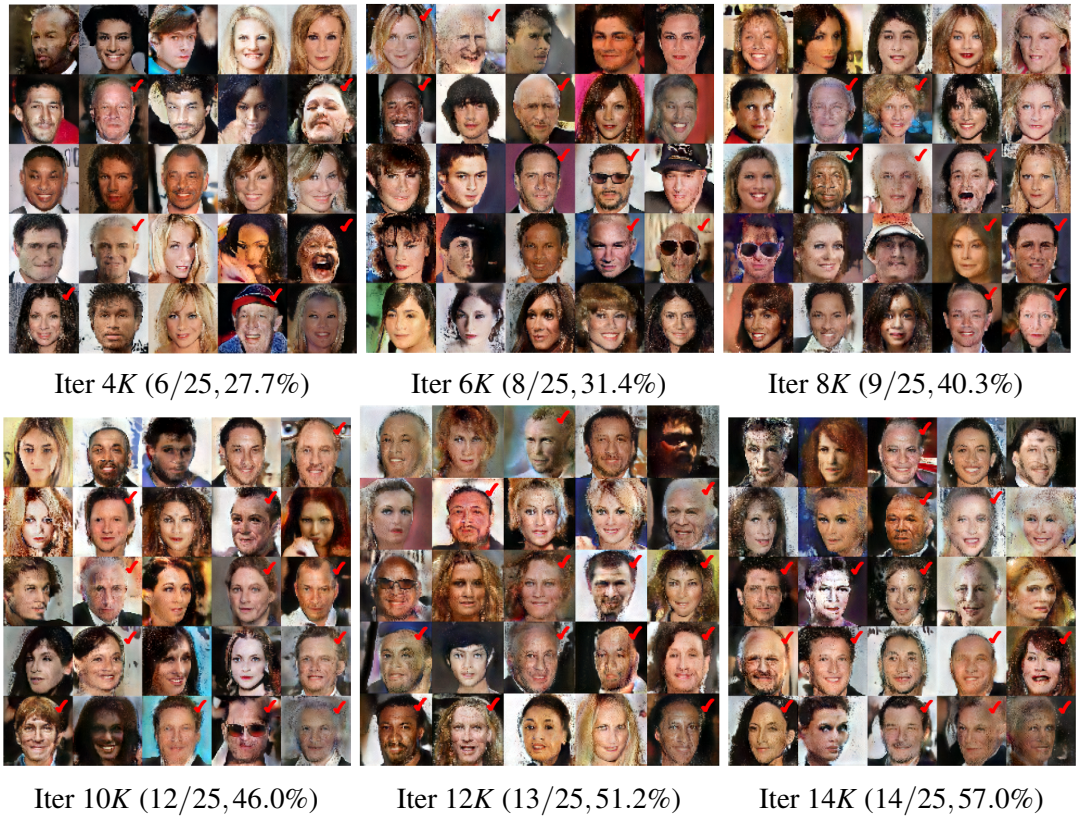


Figure 4.11 Generated images of DiCGAN$_{\text{style}}$ on CelebA-HQ during the training process. DiCGAN$_{\text{style}}$ learns the distribution of old faces, which gradually generates more old face images. The red ticks refer to the images which are classified as old images. The % denotes the percentage of old faces in $50K$ generated samples.

DiCGAN achieves the best image quality among all the methods. Its FID score is the lowest. In addition, DiCGAN shifts more towards the desired data distribution than CWGAN and FBGAN, proven by a larger PDD.

On the other hand, WGAN-GP architecture is limited to approximating the complex distribution of CelebA-HQ data and thus cannot generate images with very high quality. Then, the introduction of generated samples into the training data will degrade the quality of generation. Therefore, DiCGAN implemented with WGAN-GP architecture is restricted with certain amounts of minor corrections and data replacement in order to obtain a good quality, achieving relatively low PDD. This problem can be improved by introducing a more advanced GAN architecture, StyleGAN. DiCGAN$_{\text{style}}$ can conduct more minor corrections and use more generated data to replace the training data, finally making the training data distribution shift very close to the desired data distribution. Thus, DiCGAN$_{\text{style}}$'s generation contains more desired samples than DiCGAN, i.e., larger PDD in Table 4.3. Meanwhile, the generation has good quality with the best FID. We present generated images of DiCGAN$_{\text{style}}$

Figure 4.12 Nearest neighbors of generated desired images in the training dataset. The distance is measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples generated by our DiCGAN. Images on the right are top 5 nearest neighbors in the training dataset.

during the training process in Fig. 4.11. There gradually appears more desired face images, i.e., old face images in DiCGAN$_{\text{style}}$'s generation.

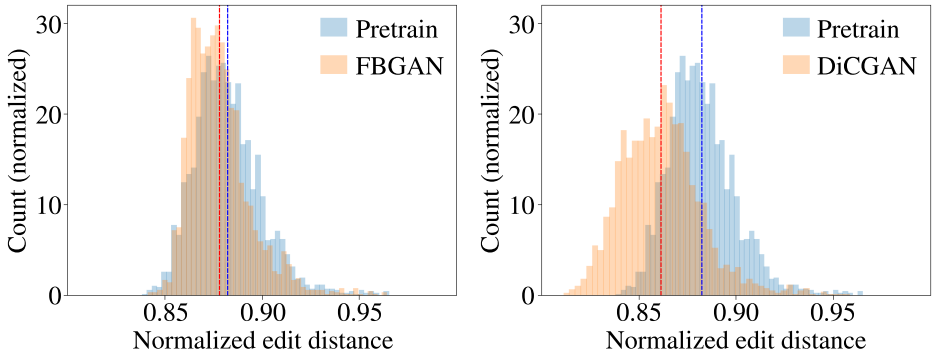
Fig. 4.12 shows the nearest neighbors of generated old images in the training dataset (given old face images), which demonstrates that our DiCGAN is not simply memorizing training images, but generates novel desired images. Thus, DiCGAN can perform data augmentation for desired samples.

4.7.3 Simulating Synthetic Genes with Antimicrobial Properties

Consider the biologist is interested in designing genes coding for antimicrobial peptides (AMPs), which are peptides with broad antimicrobial activity against bacteria, viruses, and fungi [Izadpanah and Gallo, 2005]. We can apply our DiCGAN to help optimize the gene coding for AMPs from an existing gene sequence dataset [Gupta and Zou, 2019]. Namely, our target is to learn the distribution of genes coding for AMPs on the gene sequence dataset.

Networks & Hyperparameters Both the balance factor λ and the ranking margin m are set to 1. The batch size b is set to 64. All methods are implemented with the networks as FBGAN [Gupta and Zou, 2019]. The code of the network architecture can be found on FBGAN's official implementation (*https://github.com/av1659/fbgan*). The networks are optimized with Adam with a learning rate of $1e$-4 and $\beta_1 = 0.5, \beta_2 = 0.9$. n_{critic} is set to 10.

Training There is no labeling about whether the genes have the desired property in the training data. Therefore, WGAN, CWGAN, and GAN-FT cannot be applied. Following



(a) FBGAN (b) DiCGAN

Figure 4.13 Comparison of (a) FBGAN and (b) DiCGAN on the gene sequence dataset. The dashed line denotes the mean value. The normalized edit distance is calculated between synthetic proteins and real desired proteins. A smaller distance denotes the generated genes are more similar to the desired genes.

FBGAN [Gupta and Zou, 2019], we resort to an analyzer that can evaluate the property for genes. We pertain FBGAN and DiCGAN as vanilla WGAN using $3,655$ gene sequences. Then, we train FBGAN and DiCGAN with $2K$ gene sequences and collect the results for each method. Here we limit the amount of querying the analyzer to $6K$. n_{minor} is set 31. n_{g} is set to 500.

Sample selection in FBGAN The selector in FBGAN selects desired samples based on the evaluation of the analyzer, which is able to predict the probability of a gene coding for AMPs. Specifically, the analyzer first estimates the probability of generated genes coding for AMPs. Then, the generated genes with the estimated probability over 0.8, considered as the desired genes, are selected by the selector to replace the old training data.

Pairwise preferences construction in DiCGAN We consider the analyzer [1] as the user, where a larger predicted value denotes that the gene is preferred for coding AMPs. Then, the pairwise comparison can be obtained for a pair of samples x_1 and x_2 according to their predicated values $p(x_1)$ and $p(x_2)$, i.e., $x_1 \succ x_2$ if $p(x_1) > p(x_2)$, and vice versa. At each iteration, n_s is set to 64. n_{minor} is set to 31.

The difference between genes is evaluated via the Normalized Edit Distance (normalized Levenstein distance, NED) between the proteins coded by the corresponding genes [Gupta and Zou, 2019]. The similarity of a generated gene to the desired gene can be evaluated using the averaged NED between its corresponding protein and all real desired proteins (i.e., AMPs). Particularly, a smaller NED w.r.t. AMPs denotes the generated genes more similar to genes coded for AMPs.

[1] Or we can ask biological experts to compare pairs of samples in terms of the desired property if the desired properties cannot be expressed objectively.

In Fig. 4.13, we compare DiCGAN and WGAN w.r.t. the NED between AMPs and the synthetic proteins. It shows that the synthetic proteins of DiCGAN shift toward a lower edit distance from AMPs, compared to the pretraining stage, i.e., WGAN. It means more genes coded for AMPs are generated by DiCGAN. However, FBGAN fails to shift its distribution towards the distribution of genes coding for AMPs.

Table 4.4 Percentage of desired data in the generation (PDD) and gene quality (%VG) of various GANs on the gene sequence dataset. The best results are highlighted in bold.

Method	FBGAN	DiCGAN
PDD	29.0	**98.8**
%VG	57.8	**67.0**

Further, we sample $50K$ genes from the generator for quantitative evaluation, which is collected in Table 4.4. The generated genes with the probability of coding for AMPs over 0.8 is considered as the desired genes. Then, the percentage of the desired genes among all $50K$ generated genes (PDD) is calculated. Particularly, almost all genes generated by DiCGAN can be classified as the desired genes, i.e., 98.8%. In contrast, FBGAN generates 29.0% desired genes. DiCGAN can learn the distribution of desired genes. However, FBGAN fails to derive the desired data distribution due to limited supervision.

On the other hand, we calculate the percentage of valid genes (%VG) [2]. The %VG of DiCGAN is 67.0% while that of FBGAN is 57.8%, which clarifies our methods achieve better quality than FBGAN. Their quality degradation compared to pre-trained WGAN is due to the introduction of generated genes as training data.

4.7.4 Study on critic values versus user preferences

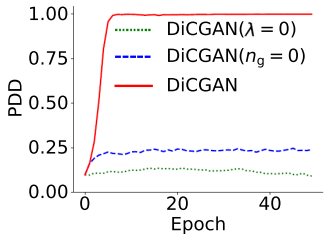
We apply DiCGAN to evaluate the critic values for all undesired data and all desired data on MNIST, CelebA-HQ, and the gene sequence dataset, respectively. Then we calculate the mean critic values for desired data and undesired data, respectively, with 95% confidence interval. Meanwhile, we conduct the two-sample one-sided t-Test [Welch, 1947] for their mean critic values under the null hypothesis of equal means and the alternative hypothesis that the mean of the desired data is greater than that of the undesired data.

The results in Table 4.5 show that the average critic value for desired data is significantly larger than that of undesired data with a very small p-value. Therefore, it verifies the claim

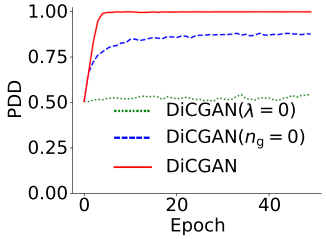
[2]Correct gene structure is defined as a string starting with the canonical start codon “ATG”, followed by an integer number of codons of length 3, and ending with one of three canonical stop codons (“TAA”, “TGA”, “TAG”) [Gupta and Zou, 2019].

Table 4.5 The mean (with 95% confidence interval) and the two-sample one-sided t-Test results of critic values for desired data and undesired data on MNIST, CelebA-HQ, and the gene sequence dataset.

Dataset	mean critic value		p value
	desired	undesired	(desired vs. undesired)
MNIST	1.5 ± 0.01	0.2 ± 0.01	0.00
CelebA-HQ	0.9 ± 0.02	0.4 ± 0.01	2.58e-285
gene	8.9 ± 0.14	8.1 ± 0.05	3.54e-24



(a) top 1



(b) top 5

Method	top 1	top 5
Original	9.9	51.1
DiCGAN ($\lambda = 0$)	9.1	54.6
DiCGAN ($n_g = 0$)	24.0	87.8
DiCGAN	**100.0**	**100.0**

(c) PDD

Figure 4.14 Ablation study on MNIST. (a-b) PDD vs. epoch in the generation of DiCGAN ($\lambda = 0$), DiCGAN ($n_g = 0$) and DiCGAN. (c) PDD in the data from the original dataset, DiCGAN ($\lambda = 0$), DiCGAN ($n_g = 0$) and DiCGAN.

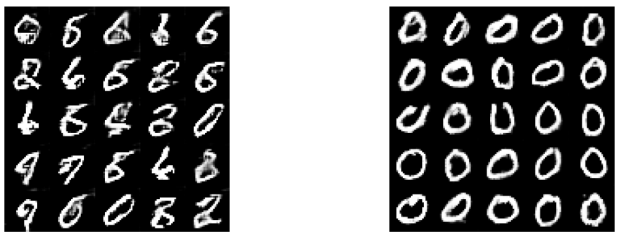
(mentioned in Sect. 4.2) that the ranking loss can encourage high critic values to be assigned to the real desired data while low critic values are assigned to real undesired data.

4.7.5 Ablation Study

The objective in our DiCGAN (Eq. (4.3)) consists of two components, i.e., the WGAN loss, which serves as the cornerstone of DiCGAN, and the ranking loss, which serves as the correction to WGAN. Meanwhile, we introduce the operation of replacement (Eq. (4.7)) during the model training.

To analyze the effects of the correction for WGAN (the third term in Eq. (4.3)) and the replacement operation, we plot the percentage of desired data in the generation (PDD) versus the training epoch for DiCGAN ($\lambda = 0$), DiCGAN ($n_g = 0$) and DiCGAN in Fig. 4.14a, 4.14b. Meanwhile, the converged percentage of desired samples (PDD) is reported in Fig. 4.14c.

1) **Without the correction term ($\lambda = 0$), DiCGAN cannot learn the desired data distribution.** The PDD of DiCGAN ($\lambda = 0$) remains constant during training on MNIST



(a) PRG-1 (PDD=13.9) (b) PRG-2 (PDD=99.4)

Figure 4.15 Generated digits and PDD of (a) PRG-1 & (b) PRG-2.

(Fig. 4.14a, 4.14b) compared with that of the original dataset (Fig. 4.14c). This is because the WGAN term in DiCGAN ($\lambda = 0$) focuses on learning the training data distribution.

2) **Without the replacement ($n_g = 0$), DiCGAN makes a minor correction to the generated distribution.** In Fig. 4.14a, 4.14b, the PDD of DiCGAN ($n_g = 0$) slightly increases compared with the original dataset. This is consistent with our analysis that the correction term would drive the generation towards the desired data distribution.

3) **DiCGAN learns the desired data distribution with a sequential minor correction.** The PDD of DiCGAN grows with training and reaches almost 100% when convergence. The correction term drives DiCGAN's generation towards the desired data slightly at each epoch. With the iterative replacement, the minor correction sequentially accumulates, and finally the generated distribution shifts to the desired data distribution.

4.7.6 Pairwise regularization on the Generator

As discussed in Sect. 4.5.2, the pairwise regularization is possibly added to the generator. We consider two cases of adding the regularization to the generator. First, we only add the pairwise regularization to the generator (PRG-1). Second, we add the regularization to the generator together with the regularization on the critic (PRG-2).

We conducted experiments on MNIST to show the effectiveness of these two methods. λ and λ' are both set to 1. As shown in Fig. 4.15, PRG-1 failed to learn the desired data distribution. PRG-2 can learn the desired data distribution. The quantitative results are consistent with the visual results, with 13.9% and 99.4% PDD, respectively.

4.8 Summary

We propose DiCGAN to learn the distribution of the user-desired data from the entire dataset using the pairwise preferences. This is the first work to promote the ratio of the desired data

by incorporating user preferences directly into the data generation, thanks to our proposed new insight for critic value (discussed in Section 4.5.1 in detail). We empirically demonstrate the efficacy of DiCGAN in two real-world applications – generating images that meet the user's interest for a given dataset and optimizing biological products with desired properties. Especially, our DiCGAN outperforms baselines in the cases of insufficient desired data and limited supervision.

Though it is superior to existing methods in terms of desired data generation when there is insufficient desired data, our DiCGAN cannot handle the case when there are extremely limited desired data, e.g., few-shot even one shot. This is because the distribution of desired data cannot be accurately modeled in such a scenario. Furthermore, as shown in our experimental study, high-resolution high-quality desired image generations require an advanced GAN architecture, which incurs heavy computational costs. There is an ongoing research direction of GAN that aims to generate high-resolution high-quality data with light architecture designs, which can mitigate such a limitation. Last, the idea of DiCGAN can also be extended to other GAN variants that relies on a critic, such as RGAN [Jolicoeur-Martineau, 2019].

Chapter 5

Generative Adversarial Ranking based on Preferences

As demonstrated in Section 4.3, DiCGAN relies on multi-step distribution shifts to converge to desired data distribution, which leads to costly annotation on generated samples. In this chapter, we propose a new generative modeling paradigm to learn the desired data distribution from human preferences in single step. To be specific, we propose a new adversarial training framework – generative adversarial ranking network (GARNet) to learn from human preferences among a list of samples so as to generate data meeting user-specific criteria. Verbosely, GARNet consists of two modules: a ranker and a generator. The generator fools the ranker to raise generated samples to the top; while the ranker learns to rank generated samples at the bottom. Meanwhile, the ranker learns to rank samples regarding the interested property by training with preferences collected on real samples. The adversarial ranking game between the ranker and the generator enables an alignment between the generated data distribution and the desired data distribution with theoretical guarantees and empirical verification. Specifically, we first prove that when training with full preferences on a discrete property, the learned distribution of GARNet rigorously coincides with the distribution specified by the given score vector based on user preferences. The theoretical results are then extended to partial preferences on a discrete property and further generalize to preferences on a continuous property. Meanwhile, numerous experiments show that GARNet can retrieve the distribution of user-desired data based on full/partial preferences in terms of various interested properties (i.e., discrete/continuous property, single/multiple properties).

5.1 Problem Statement

Let $\mathcal{X} = \{x_n\}_{n=1}^{N}$ denote a training dataset with N samples, $\mathcal{S} = \{s_m\}_{m=1}^{M}$ denote a collection of M preferences defined on subsets of $\mathcal{X}$. In particular, each $s \in \mathcal{S}$ is an ordered list, namely,

$$s := s^1 > s^2 > \ldots > s^l, \text{ and } s^i \in \mathcal{X}, \forall i = 1, 2, \ldots, l, \tag{5.1}$$

where $2 \leq l \ll N$ denotes the number of samples contained in s. l is usually different for different preferences. Our target is to learn a generative model $P_{\mathrm{g}}(x)$ that is equal to the desired data distribution $P_{\mathrm{d}}(x)$ conditioned on partial preferences $\mathcal{S}$. To be specific, $P_{\mathrm{d}}(x)$ should allocate high density to global high-ranked data while low or even zero density to global low-ranked data[1].

Remark 3. *The problem defined above is similar as that in Chapter 4 (Def. 2). But differently, the given preferences are listwise instead of pairwise. In addition, the target distribution is a generalized user-desired data distribution that has fine-grained density for data with different preferences.*

For example, as shown in Fig. 5.1, we assume that shoe images with different strengths in terms of the *open* attribute are distributed evenly. The collected pairwise preferences reveal that users prefer more *open* shoes. Accordingly, the generative model is expected to learn a distribution that assigns its density on shoe images with large *open* values. Note that the generative model has the advantage of generating novel images different from the training data [Song and Ermon, 2019].

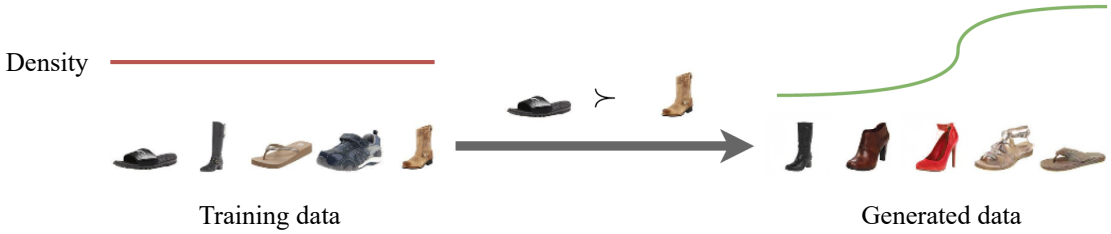


Figure 5.1 Illustration of desired data distribution learning guided by users' preferences (on *open* attribute).

[1] User preferences help to derive a global ranking over all the data [Cao et al., 2007b, Lu and Boutilier, 2011]. Global high-ranked data denote those data ranked high in the global ranking. Global low-ranked data denote those data ranked low in the global ranking.

5.2 GARNet for Preferences w.r.t. a Discrete Property

First of all, we consider a scenario in which the ranking relation among training samples is based on a discrete property. In particular, each training sample belongs to one of a finite set of classes which possess a natural order.

Definition 3 (Preferences w.r.t. a discrete property)**.** *Let $\mathcal{X} = \{x_n\}_{n=1}^{N}$ be a set of training instances and $\mathcal{Y} = \{y_1, y_2, \ldots, y_T\}$ be a set of class labels endowed with an order $y_1 > y_2 > \ldots > y_T$, where T is the number of ordered classes. Assume each training instance x_i is assigned with a label $y(x_i) \in \mathcal{Y}$, where $y(\cdot)$ is a function that outputs the label for a sample. The preference over a group of samples $\{x_{i_1}, x_{i_2}, \ldots,\} \subseteq \mathcal{X}$ is defined as*

$$\begin{aligned} x_{i_1} > x_{i_2} > x_{i_3} > \ldots, \\ \textit{if} \quad y(x_{i_1}) > y(x_{i_2}) > y(x_{i_3}) > \ldots. \end{aligned} \tag{5.2}$$

We ignore the preference lists containing samples with the same labels since such preference lists can be simply transformed into the preferences (Eq. (5.2)) by removing redundant samples [Pan et al., 2022].

Example. Suppose users prefer small digits on the MNIST dataset. Then for digits 0 to 9, the order would be $0 > 1 > 2 > \ldots > 9$, according to which users express their preferences over MNIST images.

In the following part of this section, we propose a framework called generative adversarial ranking net (GARNet) for the distribution learning guided by the above-mentioned preferences (problem definition in Section 5.1). Our GARNet consists of a ranker R (parameterized by θ_R) and a generator G (parameterized by θ_G). In particular, an adversarial ranking process is defined between the generator and the ranker. Namely, they are trained against the goal of rankings defined between the real data and the generated data (See Fig. 5.2). The competition between the generator and the ranker drives the generated distribution $P_{\mathrm{g}}(x)$ (simplified as P_{g}) aligned with the desired data distribution $P_{\mathrm{d}}(x)$ (simplified as P_{d}).

Remark 4. *It is impossible to manipulate PDF of different classes based on preferences. The class label information for the data is inaccessible since our target problem is given partial/full preferences.*

5.2.1 Given Full Preferences

We open the description of our GARNet with the case given a collection of full preferences $\mathcal{S} = \{s_m\}_{m=1}^{M}$. That is, the length of each preference l is equal to the number of ordered

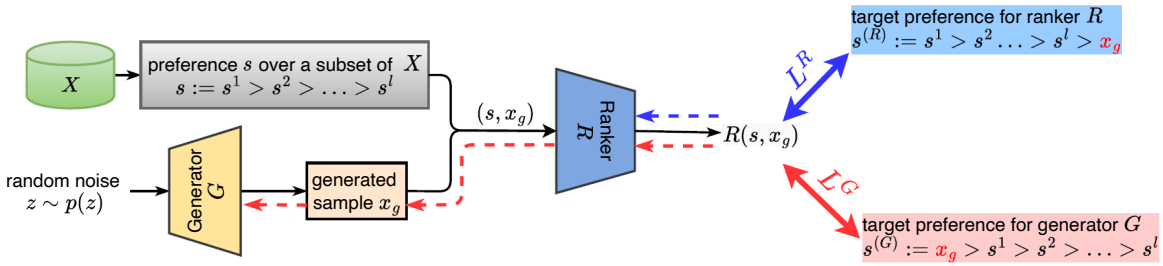


Figure 5.2 Framework of GARNet.

classes T. Namely

$$s := s^1 > s^2 \ldots > s^l,\ l = T \text{ where } s^i \in \mathcal{X},\ \forall i = 1,2,\ldots,l. \tag{5.3}$$

A full preference s would include samples from all ordered classes. Consequently,

$$y(s^i) \equiv y_i,\ \forall s \in \mathcal{S} \text{ and } \forall i = 1,2,\ldots,T. \tag{5.4}$$

Denoting P_i as the distribution of data $\mathcal{X}_i = \{x | y(x) = y_i, x \in \mathcal{X}\}$, then we have $s^i \sim P_i$.

Listwise Ranking

The ranker R learns a ranking function from the preferences $\mathcal{S}$. We employ ListNet [Cao et al., 2007b], which is a cross entropy based learning-to-rank loss and can handle preference lists that are not equally informative by assigning different losses according to their degree of agreement with the consensus global ranking [Lin, 2010]:

$$\mathcal{L}_{CE}\left(\pi(s), R(s)\right) = -\sum_{i=1}^{l} \sigma(\pi(s))_i \log \sigma(R(s))_i. \tag{5.5}$$

With a slight abuse of notation, $s = (s^1, s^2, \ldots, s^l)$. $\pi(s)$ denotes the ground-truth score vector for the preference s, which can be explicitly or implicitly given by humans [Cao et al., 2007b]. $R(\cdot)$ denotes the score vector for the preference s predicted by the ranker R. In specific, the ranker R acts as a nonlinear feature extractor with a scalar output. Given feature matrix $w = [w_1, w_2, \ldots, w_l]$ of l samples, the ranker R outputs a score vector $R(w) = [R(w_1), R(w_2), \ldots, R(w_l)]$, where $R(w_i) \in \mathbb{R}$. $\sigma(\cdot)$ is a softmax function that takes a list of scalar values $r = [r_1, r_2, \ldots, r_l]$ as input, i.e., $\sigma(r)_i = \frac{e^{r_i}}{\sum_{j=1}^{l} e^{r_j}}$. $\sigma(\pi(s))_i$ calculates the top-1 probability of object s^i, i.e., the probability of s^i being ranked on the top given the scores

of all the objects $\pi(s)$. Similarly, $\sigma(R(s))_i$ gives the top-1 probability of object s^i given the score vector $R(s)$.

Generative Adversarial Ranking

Given a preference s and a generated sample $x_g = G(z) \sim P_g(x)$, where $z \sim \mathcal{Z}$ is a random noise, the ranker R incorporates the generated sample x_g into the ranking list s and outputs a new ranking score vector as $R(s, x_g)$. Motivated by vanilla GAN [Goodfellow et al., 2014] which designs a target real/fake label for each generated sample to trigger the adversarial game between the discriminator and the generator, we construct a target preference for a generated list of samples consisting of s and x_g, i.e., (s, x_g), as adversarial supervision. To promote the competition between the ranker R and the generator G, we design the target preference in two different ways, namely,

$$\text{target preference for } R \quad s^{(R)} := s^1 > \ldots > s^T > x_g, \tag{5.6a}$$

$$\text{target preference for } G \quad s^{(G)} := x_g > s^1 > \ldots > s^T. \tag{5.6b}$$

The generator fools the ranker to grade the generated samples as the best (Eq. (5.6b)); while the ranker learns to rank them at the lowest position (Eq. (5.6a)). Then, we define the objective for the ranker R and the generator G as follows, respectively,

$$\sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S}\\ x_g\sim P_g}} \left[\mathcal{L}_{CE}\left(\pi\left(s^{(R)}\right), R\left(s^{(R)}\right)\right)\right], \tag{5.7a}$$

$$\sup_{G:\mathcal{Z}\to\mathcal{X}} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S}\\ x_g\sim P_g}} \left[\mathcal{L}_{CE}\left(\pi\left(s^{(G)}\right), R\left(s^{(G)}\right)\right)\right]. \tag{5.7b}$$

Thus, the training loss for the ranker and the generator with the mini-batch data can be formulated as follows, respectively:

$$L^R = -\sum_{i=1}^{B} \mathcal{L}_{CE}\left(\pi\left(s_i^{(R)}\right), R\left(s_i^{(R)}\right)\right), \tag{5.8a}$$

$$L^G = -\sum_{i=1}^{B} \mathcal{L}_{CE}\left(\pi\left(s_i^{(G)}\right), R\left(s_i^{(G)}\right)\right), \tag{5.8b}$$

where B is the batch size. The training algorithm is summarized in Algorithm 3.

In most GAN variants, the objective function of the discriminator at optimum is a divergence [Jolicoeur-Martineau, 2020, Goodfellow et al., 2014, Arjovsky et al., 2017b]. In the following, we prove that training the ranker in GARNet (Eq. (5.7a)) is equivalent

Algorithm 3 Generative Adversarial Ranking Net

1: **Input:** Training data $\mathcal{X} = \{x_n\}_{n=1}^{N}$, user preferences $\mathcal{S} = \{s_m\}_{m=1}^{M}$, batch size B, ranker R and generator G.
2: **Output:** Generator G for desired data distribution, i.e., $P_{\mathrm{g}}(x) = P_{\mathrm{d}}(x)$.
3: **repeat**
4: Sample a mini-batch preferences $\{s_i\}_{i=1}^{B}$ from $\mathcal{S}$.
5: Get fake samples $\{x_{\mathrm{g}i}\}_{i=1}^{B}$ from the generator G, i.e., $x_{\mathrm{g}i} = G(z_i)$ where z_i is a random noise.
6: Following Eq. (5.6a), construct preferences $\{s_i^{(R)}\}_{i=1}^{B}$ for the ranker R.
7: Train the ranker R according to Eq. (5.8a).
8: Following Eq. (5.6b), construct preferences $\{s_i^{(G)}\}_{i=1}^{B}$ for the generator G.
9: Train the generator G according to Eq. (5.8b).
10: **until** convergence

to estimating a divergence between the desired data distribution $P_{\mathrm{d}}(x)$ and the generated data distribution $P_{\mathrm{g}}(x)$. The generator in GARNet (Eq. (5.7b)) is trained to minimize the divergence so as to achieve $P_{\mathrm{g}}(x) = P_{\mathrm{d}}(x)$.

Definition 4 (Divergence)**.** *Let $P \in \mathbb{S}$ and $Q \in \mathbb{S}$ be probability distributions where $\mathbb{S}$ is the set of all probability distributions with common support. A function $D : (\mathbb{S}, \mathbb{S}) \to [0, +\infty)$ is a divergence if it respects the following two conditions:*

$$\begin{aligned} &D(P,Q) \geq 0, \\ &D(P,Q) = 0 \Longleftrightarrow P = Q. \end{aligned} \tag{5.9}$$

Theorem 2 (Relativistic f-divergence [Jolicoeur-Martineau, 2020])**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be a concave function such that $f(0) = 0$, f is differentiable at 0, $f'(0) \neq 0$, $\sup_v f(v) = M > 0$, and $\arg\sup_v f(v) > 0$. Assume P and Q are two distributions with support $\mathcal{X}$, we have that*

$$\mathrm{D}_f(P,Q) = \sup_{C:\mathcal{X}\to\mathbb{R}} 2\,\mathop{\mathbb{E}}_{\substack{x\sim P \\ \tilde{x}\sim Q}}[f(C(x) - C(\tilde{x}))] \tag{5.10}$$

is a relativistic f-divergence.

Theorem 2 demonstrates the optimal discriminator in relativistic GAN [Jolicoeur-Martineau, 2019] estimates a relativistic f-divergence between the real data distribution and the generated data distribution. Its discriminator that is adapted from the concave function $f(z) = \log(\mathrm{sigmoid}(z)) + \log(2)$ defines a pairwise ranking loss between a real sample and a fake

sample, and we find that

$$\begin{aligned} f\left(C(x)-C(\tilde{x})\right) &= \log\left(\frac{1}{1+e^{-(C(x)-C(\tilde{x}))}}\right)+\log(2) \\ &= \log\left(\frac{e^{C(x)}}{e^{C(x)}+e^{C(\tilde{x})}}\right)+\log(2) \\ &\overset{①}{=} \mathcal{L}_{L2R}(\pi(s),C(s))+\log(2), \end{aligned} \tag{5.11}$$

① is valid if $\pi(s)=[0,-\infty]$ and $s:=x>\tilde{x}$ where $x\sim P$ and $\tilde{x}\sim Q$.

Inspired by the connection between the relativistic f-divergence (Eq. (5.10)) and the tailored loss of our GARNet (Eq. (5.6a), Eq. (5.7a)), we conjecture that the optimal ranker of our GARNet also implicitly estimate a relativistic f-divergence, but between the desired data distribution and the generated data distribution. Accordingly, we introduce a new relativistic f-divergence between the desired data distribution P_{d} and the generated data distribution P_g which generalizes the targeted data distribution $P(x)$ in Theorem 2 to the mixture distribution $P_{\mathrm{d}}=\sum_{i=1}^{T} q_i P_i$[2] with a user-specified mixing ratio where $q_1>q_2>\ldots>q_T$[3] and $\sum_i q_i=1$.

Theorem 3 (Relativistic f-divergence between P_{d} and P_g)**.** *Let $f:\mathbb{R}\to\mathbb{R}$ be a concave function such that $f(0)=0$, f is differentiable at 0, $f'(0)\neq 0$, $\sup_v f(v)=M>0$, and $\arg\sup_v f(v)>0$. Let P_{d} be the mixture distribution of the whole ordered data, i.e., $P_{\mathrm{d}}=\sum_{i=1}^{T} q_i P_i$, where $q_1>q_2>\ldots>q_T$ and $\sum_i q_i=1$. Let P_{g} be the distribution of generated data x_{g}. We have that*

$$\mathrm{D}_f(P_{\mathrm{d}},P_g)=\sup_{C:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f\left(\sum_{i=1}^{T} q_i R(s^i)-R(x_{\mathrm{g}})\right)\right] \tag{5.12}$$

is a relativistic f-divergence between P_{d} and P_g.

According to Definition 4, we prove that $D_f(P_{\mathrm{d}},P_{\mathrm{g}})$ is a divergence between P_{d} and P_{g} following the three steps:
#1 Prove that $D_f(P_{\mathrm{d}},P_{\mathrm{g}})\geq 0$.
#2 Prove that $P_{\mathrm{d}}=\sum_{i=1}^{T} q_i P_i=P_{\mathrm{g}} \Longrightarrow D_f(P_{\mathrm{d}},P_{\mathrm{g}})=0$.
#3 Prove that $D_f(P_{\mathrm{d}},P_{\mathrm{g}})=0 \Longrightarrow P_{\mathrm{d}}=\sum_{i=1}^{T} q_i P_i=P_{\mathrm{g}}$.
The details are left in Appendix C.

[2] P_i is the distribution of $\mathcal{X}_i=\{x|y(x)=y_i, x\in\mathcal{X}\}$.
[3] User-desired data distribution is considered to allocate a larger density to the higher-ranked class.

Proposition 5. *Given the training dataset $\mathcal{X} = \{x_n\}_{n=1}^N$ and a collection of full preferences $\mathcal{S} = \{s_m\}_{m=1}^M$, where each preference $s := s^1 > s^2 \ldots > s^T \in \mathcal{S}$ and $s^i \sim P_i$. let $P_\mathrm{d} = \sum_{i=1}^T q_i P_i$ be the mixture distribution of the whole ordered data where $q_i = \sigma\left(\pi\left(s^{(R)}\right)\right)_i$. Let P_g be the distribution of generated data x_g. Given a fixed generator G, the optimal ranker R^* of our GARNet (Eq.* (5.7a)*) approximates the relativistic f-divergence between P_d and P_g, i.e., $D_f(P_\mathrm{d}, P_\mathrm{g})$, if $\pi\left(s^{(R)}\right) = [a+(T-1)d, a+(T-2)d, \ldots, a, b]$ and $\frac{1}{e^{a-b}} \to 0$*[4].

The detailed proof is left in Appendix C.

Remark 5. *In theory, it seems feasible to directly adopt the following objective to learn P_d according to Theorem 3.*

$$\sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1 \\ \cdots \\ s^T\sim P_T \\ x_\mathrm{g}\sim P_\mathrm{g}}} \left[f\left(\sum_{i=1}^T q_i R(s^i) - R(x_\mathrm{g}) \right) \right], \tag{5.13a}$$

$$\sup_{G:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1 \\ \cdots \\ s^T\sim P_T \\ x_\mathrm{g}\sim P_\mathrm{g}}} \left[f\left(R(x_\mathrm{g}) - \sum_{i=1}^T q_i R(s^i) \right) \right], \tag{5.13b}$$

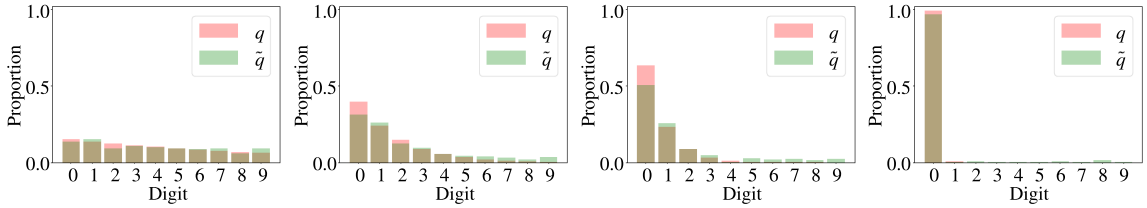
where the concave function f can be defined as $f(z) = \log(\mathrm{sigmoid}(z)) + \log(2)$ like Jolicoeur-Martineau [2019]. However, in practice, the above objective cannot provide sufficient gradient for the generator G to learn well. In the early training, the generator G is poor, and the ranker R can easily assign lower values to the generated sample x_g than real samples. $\sum_{i=1}^T q_i R(s^i) - R(x_\mathrm{g})$ would be large and then gradient vanish occurs.

Corollary 4. *The optimal ranker of GARNet (Eq.* (5.7a)*) defines a relativistic f-divergence between the distribution of top-1 data P_1 and the generated data distribution P_g, i.e., $D_f(P_1, P_\mathrm{g})$ when $e^{-d} \to 0$.*

Proof. According to the definition of q_1 in Theorem 5, we have

$$\begin{aligned} q_1 &= \frac{e^{a+(T-1)d}}{\left(\sum_{i=1}^T e^{a+(T-i)d}\right) + e^b} \\ &= \frac{e^{a+(T-1)d}}{e^{a+(T-1)d}\left(1 + e^{-d} + \ldots + e^{-(T-1)d} + e^{b-(a+T-1)d}\right)} \\ &= \frac{1}{1 + e^{-d} + \ldots + e^{-(T-1)d} + e^{b-(a+T-1)d)}} > \frac{1}{1+Te^{-d}} \end{aligned} \tag{5.14}$$

[4] In practice, when $a - b \geq 10$, $\frac{1}{e^{a-b}} \approx 0$. Thus, we set $a - b = 10$ in our experiments.



(a) $d = 0.1$ (b) $d = 0.5$ (c) $d = 1$ (d) $d = 5$

Figure 5.3 GARNet learns the distribution P_{d} with different pre-specified score vectors $\pi(s)$ from full preferences w.r.t. a discrete property on MNIST (preferring small digits, i.e., $0 \succ 1 \ldots \succ 9$). $q = \sigma(\pi(s))$ calculates the ground-truth top-1 probabillity of each digit class. $\tilde{q}$ calculates the proportion of different digit classes for generated data from GARNet. The digit values for the generated data are evaluated by a pretrained classifier for digit classification.

Since $\frac{1}{1+Te^{-d}} < q_1 < 1$, we have $q_1 \to 1$ following the Squeeze theorem [Stewart et al., 2020] when $e^{-d} \to 0$. The proof is completed. □

Proposition 6. *Given the optimal ranker R^* as demonstrated in Theorem 5, the generator of GARNet (Eq.* (5.7b)*) is minimizing the divergence between P_{d} and P_{g}, i.e., $D_f(P_{\mathrm{d}}, P_{\mathrm{g}})$.*

Proof. Given R^*, the objectives for the generator can be formulated as:

$$\sup_{G:\mathcal{Z}\to\mathcal{X}} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S} \\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\mathcal{L}_{L2R}\left(\pi\left(s^{(G)}\right), R^*\left(s^{(G)}\right)\right)\right]. \tag{5.15}$$

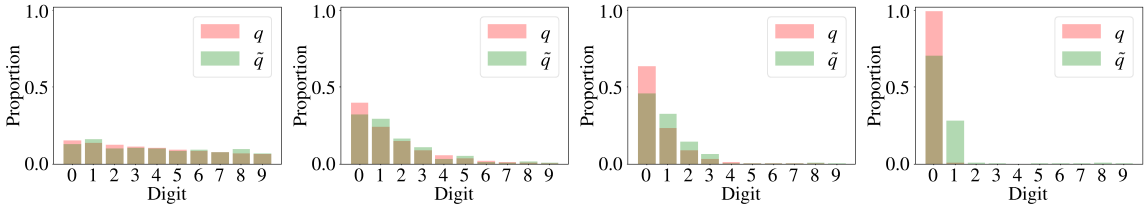
We update the parameters of generator θ_G as follows:

$$\theta_G^{t+1} = \theta_G^t + \nabla_{\theta_G} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S} \\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\mathcal{L}_{L2R}\left(\pi\left(s^{(G)}\right), R^*\left(s^{(G)}\right)\right)\right], \tag{5.16}$$

which maximizes the ranker's output for the generated samples $R^*(x_{\mathrm{g}})$. Then, $D_f(P_{\mathrm{d}}, P_{\mathrm{g}})$ will be minimized. □

Proposition 5 and Proposition 6 demonstrate that the distribution learned by our GARNet (Eq. (5.7)) is determined by the pre-specified score vector $\pi(s)$. We present a case study on MNIST [Lecun et al., 1998] to justify it by setting d to $0.1, 0.5, 1, 5$, respectively. As shown in Fig. 5.3, the proportion of generated data from GARNet is almost consistent with the top-1 probability calculated by the specified score vector, i.e., $q = \sigma(\pi(s))$.

In terms of preferences over a discrete property, the most desirable data is supposed to belong to the global top-1 class, i.e., $y(x) = y_1$. As clarified in Corollary 4, we can simply set up a sufficiently large d to achieve this goal.



(a) $d' = 0.1$ (b) $d' = 0.5$ (c) $d' = 1$ (d) $d' = 5$

Figure 5.4 GARNet learns the distribution P_{d} with different pre-specified score vectors $\pi(s)$ from partial preferences ($2 \leq l \leq 10$) w.r.t. a discrete property on MNIST (preferring small digits). d' is the common difference of the ground-truth score vector $\pi\left(s^{(R)}\right)$ for Eq. (5.18a). $\tilde{q}$ calculates the proportion of different digit classes for generated data from GARNet. q is copied from the results in Fig. 5.3 for references.

5.2.2 Given Partial Preferences

In many situations, full preferences are not available. Instead, only partial preferences[5] of length l less than the number of ordered class T are accessible [Lin, 2010]. Namely,

$$s := s^1 > s^2 > \ldots > s^l, \quad l < T. \tag{5.17}$$

Similarly, we construct the adversarial ranking process between the ranker and generator:

$$\text{ranker} \qquad s^{(R)} := s^1 > \ldots > s^l > x_{\mathrm{g}}; \tag{5.18a}$$

$$\text{generator} \qquad s^{(G)} := x_{\mathrm{g}} > s^1 > \ldots > s^l. \tag{5.18b}$$

We adopt the objective (Eq. (5.7)) to learn the desired distribution $P_{\mathrm{d}}(x)$ via the adversarial ranking (Eq. (5.18)). Thanks to the consistency of the score function (Lemma 2), our GARNet can still learn the desired data distribution from the partial preferences.

Lemma 2 (Consistency of the score function [Xia et al., 2008])**.** *Given a collection of partial preferences $\mathcal{S} = \{s_m\}_{m=1}^{M}$, where $2 \leq |s_m| \leq T$, the ListNet loss (Eq. (5.5)) adopted in our GARNet can recover the optimal score function for the ranker if $\mathcal{S}$ contains sufficient preferences and the samples $\mathcal{X}$ are well covered*[6]*.*

According to Lemma 2, the optimal ranker will recover $s^1 > s^2 > \ldots > s^T > x_{\mathrm{g}}$. Therefore, GARNet can still learn a desired data distribution from partial preferences, which assigns greater density to higher-ranked data. We present a case study on MNIST [Lecun et al.,

[5] Partial preferences can be of different lengths in the training dataset.

[6] It doesn't mean to include every sample into the preferences but requires the samples included to be approximately uniformly distributed in the score space.

1998] with partial preferences ($2 \leq l \leq 10$). We set the ground-truth score vector $\pi\left(s^{(R)}\right)$ for Eq. (5.18a) as $[a+(l-1)d', a+(l-2)d', \ldots, a, b]$ similar in Proposition 5. As shown in Fig. 5.4, the proportion of generated data from GARNet is larger when the digit class is ranked higher, which is consistent with user preferences. Especially, when d' is sufficiently large, GARNet only generates high-ranked data, namely, digit zero and one. We found that training on relatively long preferences ($7 \leq l \leq 10$), GARNet can converge to top-1 data distribution when setting $d' = 5$. Note the resultant proportion per digit $\tilde{q}$ is no longer consistent with pre-specified values q as the $y(s^i) \equiv y_i$, $\forall s \in \mathcal{S}$ and $\forall i = 1, 2, \ldots, T$ does not hold anymore for partial preferences.

5.3 GARNet for Preferences w.r.t. a Continuous Property

We consider a scenario that the ranking relation among training samples is based on a continuous property. Particularly, each training sample x_i is associated with an underlying score $o(x_i)$ that that represents the user's preference for x_i in terms of the property.

Definition 5 (Preferences w.r.t. a continuous property)**.** *Given a training dataset $\mathcal{X} = \{x_n\}_{n=1}^N$, where each training instance $x_n \in \mathcal{X}$ has an underlying ground-truth ranking score $o(x_n) \in [A, B] \subseteq \mathcal{O}$ in term of a certain continuous property, the preference among a list of samples $\{x_{i_1}, x_{i_2}, \ldots, \} \subseteq X$ is defined as*

$$\begin{aligned} x_{i_1} > x_{i_2} > x_{i_3} > \ldots, \\ \text{if} \quad o(x_{i_1}) > o(x_{i_2}) > o(x_{i_3}) > \ldots. \end{aligned} \tag{5.19}$$

Example. Suppose users prefer smiling face images. Then users will assign higher preferences to those images have bigger smiles, i.e., the score for sample x in terms of the "smiling" attribute $o(x)$ is larger.

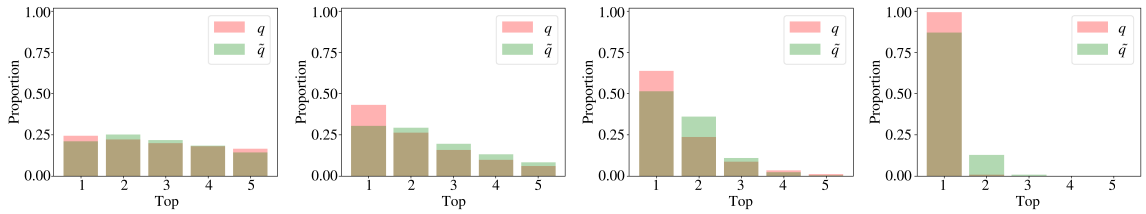
Theorem 4. *Given a collection of preferences in terms of a continuous property over $[A, B]$, see Eq.* (5.19)*, it can be transformed equivalently to the preferences in terms of finite ordered classes if we consider the samples with close ranking scores as ties.*

Proof. According to Heine-Borel theorem [Jeffreys et al., 1999], we have

$$O = \{(o_i - \varepsilon_i, o_i + \varepsilon_i) | o_i \in [A, B], \varepsilon_i > 0, i = 1, 2, \ldots, T'\}, \tag{5.20}$$

which is a finite subcover of $[A, B]$. Without loss of generality, we assume

$$o_1 > o_2 > \ldots > o_{T'}. \tag{5.21}$$



(a) $d = 0.1$ (b) $d = 0.5$ (c) $d = 1$ (d) $d = 5$

Figure 5.5 GARNet learns the distribution P_{d} with different pre-specified score vectors $\pi(s)$ from full preferences w.r.t. a continuous property on LFW (preferring smiling faces). $q = \sigma(\pi(s))$ calculates the top-1 probabillity of each digit class. $\tilde{q}$ calculates the proportion of different digit classes for generated data from GARNet. The scores for the generated data are evaluated by a pretrained ranker for ranking face images w.r.t. the *smile* attribute. Top 1 represents images ranked top 20% in terms of the *smile* attribute; top 2 represents images ranked between top 20% and top 40%; so on and so forth.

Then, we can construct the following set of internals to cover the score interval $[A, B]$:

$$O = \Big(\bigcup_{i=T',\ldots,2} [o_i - \varepsilon_i, o_i + \varepsilon_i) \Big) \cup [o_1 - \varepsilon_1, o_1 + \varepsilon_1], \tag{5.22}$$

where $o_{T'} - \varepsilon_{T'} = A, o_1 + \varepsilon_1 = B, o_i + \varepsilon_i = o_{i-1} - \varepsilon_{i-1}, i = T', \ldots, 2$.

Meanwhile, we define a pair of samples within the same internal as a tie following Zhou et al. [2008], namely

$$\begin{gathered} x_m = x_n \\ \text{for any } o(x_m), o(x_n) \in O_i, \end{gathered} \tag{5.23}$$

where $O_i = [o_i - \varepsilon_i, o_i + \varepsilon_i), i = T', T' - 1, \ldots, 2$ and $O_1 = [o_1 - \varepsilon_1, o_1 + \varepsilon_1]$.

By assigning the same label y_i to all samples within each interval O_i, we can obtain a set of ordered class labels $\{y_i\}_{i=1}^{T'}$ where $y_1 > y_2 > \ldots > y_{T'}$, namely

$$\begin{gathered} y(x_i) \equiv y_i \\ \text{if} \quad o(x_i) \in O_i. \end{gathered} \tag{5.24}$$

To sum up, the preferences in terms of a continuous property (Definition 5) approximate the preferences in terms of T' ordered classes. □

Remark 6. *The assumption of Eq.* (5.23) *is in line with real-world applications. People usually cannot distinguish a pair of samples when the difference is small and would annotate them as ties [Zhou et al., 2008], where ε_i is the indistinguishable score difference for each O_i. Nevertheless, for $o(x_1) \in O_{i-1}$ and $o(x_2) \in O_i$, x_1 and x_2 are indistinguishable when*

$o(x_1) - o(x_2)$ is small. Assigning them as two ordered classes y_{i-1} and y_i is reasonable when the order between y_{i-1} and y_i is not significant by setting T' to be sufficiently large.

According to Theorem 4, learning from the preferences in terms of a continuous property can be discretized as learning from the preferences over a finite number of ordered classes. Therefore, we can adopt GARNet which defines an adversarial ranking goal (Eq. (5.6)) to learn the desired data distribution from full preferences in terms of the continuous property. We present a case study on LFW [Huang et al., 2008] w.r.t. the *smile* attribute in Fig. 5.5 to justify this. Simply, we discrete the continuous ranking w.r.t. the *smile* attribute into five ordered classes. That is, images ranked top 20% in terms of the *smile* attribute is assigned as top 1 class; images ranked between top 20% and top 40% in terms of the *smile* attribute is assigned as top 2 class; so on and so forth. Similar in Fig. 5.3, we specify different score vectors $\pi(s)$ by setting d to $0.1, 0.5, 1$, respectively. As shown in Fig. 5.5, the proportion of generated data from GARNet is almost consistent with the top-1 probability calculated by the specified score vector, i.e., $q = \sigma(\pi(s))$.

Similarly, we can adopt GARNet (Eq. (5.18)) to learn the desired data distribution from partial preferences in term of the continuous property. Note that in practice, we do not explicitly discretize the continuous ranking scores to a finite number of classes and define a so-called top-1 class as the most desirable data. Instead, we suppose the generative model is more consistent with user preferences if the generated data has higher ranking scores.

5.4 GARNet for a Mixture of Preferences

We extend our GARNet to a more general circumstance where a mixture of user preferences are collected from distinct groups of users. Particularly, we consider a simple situation where each group of users rank the samples in terms of a specific attribute, i.e., $\{\mathcal{S}_u\}_{u=1}^U$. The objective of GARNet in this context can be formulated as follows:

$$\sup_{R:\mathcal{X}\to\mathbb{R}} \sum_{u=1}^{U} \mathop{\mathbb{E}}_{\substack{s_u\sim\mathcal{S}_u \\ x_g\sim P_g}} \left[\mathcal{L}_{CE}\left(\pi\left(s_u^{(R)}\right), R_u\left(s_u^{(R)}\right) \right) \right], \tag{5.25a}$$

$$\sup_{G:\mathcal{Z}\to\mathcal{X}} \sum_{u=1}^{U} \mathop{\mathbb{E}}_{\substack{s_u\sim\mathcal{S}_u \\ x_g\sim P_g}} \left[\mathcal{L}_{CE}\left(\pi\left(s_u^{(G)}\right), R_u\left(s_u^{(G)}\right) \right) \right]. \tag{5.25b}$$

For simplicity, we extend the scalar output of the ranker R to a U dimension vector so as to learn from $\{\mathcal{S}_u\}_{u=1}^U$ simultaneously. R_u is the u-th output of the ranker.

Conditional GARNet When provided a mixture of preferences where one group of preferences might be conflicting with another, GARNet can achieve a compromise performance, which will be empirically verified in the experiment. However, for exactly opposing attributes, like being both "open" and "not open", a practical way is to model them using two separate GARNets or extending our GARNet to its conditional version, denoted as CGARNet, conditioned on each group of preferences. The extension is similar as extending GAN to conditional GAN [Mirza and Osindero, 2014]. Both the ranker and the generator can be conditioned on an attribute label.

5.5 Discussions

In this section, we discuss how current GANs variants can be extended to preference-guided generation and their defects. To simplify the discussion, here we aim to learn the distribution that is most consistent with user preferences, i.e., the distribution of top-1 data P_1.

Preferences guided generative adversarial learning To apply vanilla GAN (denoted as GAN-0) [Goodfellow et al., 2014] for this application, an intuitive way is to first construct a dataset consisting of the user-desired samples and then perform regular GAN training on this dataset to obtain a generator that purely outputs the desired samples. Two possible strategies can be applied:

(1) *Selecting desired samples using partial preferences directly* (GAN-1): A new dataset is constructed by selecting top-1 samples from each partial preference s_m. Since the top-1 samples in partial preferences are not necessarily the global top-ranked samples, this produces biased training data that inevitably involves undesired samples. As a result, the derived distribution is not precisely user-desired. On the other hand, it would suffer from insufficient data issues when the amount of training data is small.

(2) *Selecting desired samples with a ranking proxy* (GAN-2): Specifically, we train a global ranking proxy using partial preferences [Cao et al., 2007b] and then apply the proxy to select the global high-ranked samples as a new training dataset. For example, the top 30% of the training samples are selected as the desired dataset. In spite of its feasibility, this strategy may also incur insufficient data issues especially when the desired data in the original training dataset is limited.

Feedback GAN (FBGAN) [Gupta and Zou, 2019] is based on the second strategy and remedies its drawbacks by introducing the high-ranked generated samples into previous training data. Since the ratio of the high-ranked samples gradually increases in the training data along with the training epoch, all training data will ideally converge to the user-desired

samples. However, FBGAN may suffer from severe data quality issues since plainly treating the generated samples as the training data will degrade the generation quality.

GANs plus a ranking module Another idea is to introduce a ranker as an extra critic [Saquil et al., 2018] to promote the generation of user-desired samples while the original GAN's discriminator is kept as a critic to guarantee the generation quality, which is dubbed as GAN-RK.

Similarly, the ranker learns to rank samples from partial preferences and outputs high ranking scores for user-desired samples. The discriminator can be a classifier that distinguishes real from fake [Goodfellow et al., 2014] or define a distribution discrepancy [Arjovsky et al., 2017b]. For the generator, apart from the goal of aligning the generated distribution with the real distribution guided by the discriminator, an extra goal is in pursuit of generating samples with high-ranking scores judged by the ranker. However, both goals are conflicting since one requires the generator to synthesize samples alike the whole data samples while the other requires the generation of partial data samples, i.e., samples with high-ranking scores. Thus, learning the desired data distribution cannot be well achieved.

Generator with Naive Ranker Another naive approach is to just have a ranker that learns from user preferences on real data to guide a generator, which is called GR. However, such an approach would suffer from poor data quality issues since there is no modeling for data authenticity in this model.

5.6 Experiments

We empirically verify that our GARNet can learn a distribution of high-ranked data from user preferences. Furthermore, we demonstrate the potential of GARNet in improving imbalanced class learning.

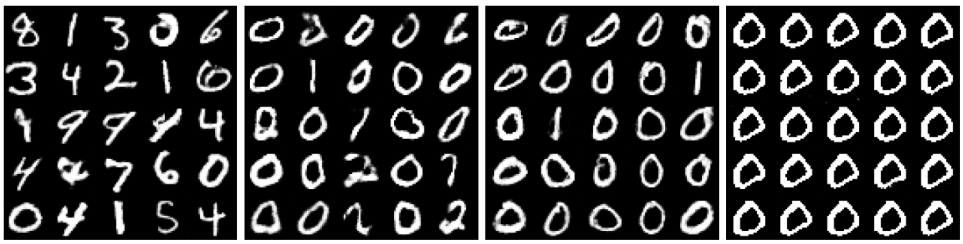
Dataset: (1) MNIST dataset [Lecun et al., 1998] consists of 28×28 images with digit zero to nine. We use its training set ($50K$ images) for experiment. We suppose that users prefer smaller digits of MNIST. An image with a smaller digit value will be ranked higher. For instance, the partial ranking list over four digits $1,3,2,9$ is $s := 1 > 2 > 3 > 9$. As there are 10 digits in total, the maximal length of the preferences l can be 10. Without loss of generality, the length of the preferences included in the training data varies from 2 to 10.
(2) Labeled Faces in the Wild (LFW) dataset [Huang et al., 2008] consists of $13,143$ celebrity face images from the wild. LFW-10 consists of a subset of $2,000$ images from LFW along with about 500 pairwise comparisons for one attribute. We take the pairwise comparisons in

terms of *smile* attribute as the user preferences for the training data. Since LFW10 has limited pairwise comparisons, we exploit a pretrained ranker to augment more preferences for the training data. In addition, as LFW10 has limited training images, i.e., $1K$ images, we take $13,143$ images from LFW as our training data also. Specifically, we pretrain a ranker to learn to rank images in terms of *smile* attribute from given pairwise comparisons of LFW10. Then we use the ranker to output ranking scores for all training samples of LFW and construct pairwise preferences based on the scores. If face image x_a has larger *smile* than face image x_b, then the preference is $s := x_a > x_b$.

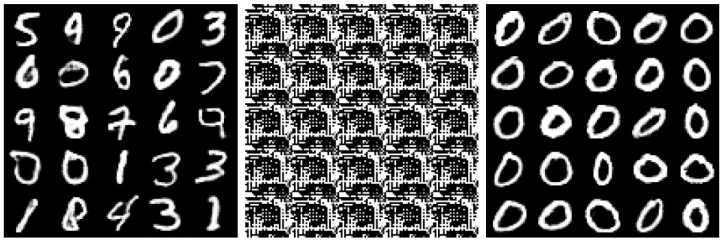
(3) UT-Zap50K dataset [Yu and Grauman, 2014] contains $50,025$ shoe images from Zappos.com. It contains pairwise comparisons over several attributes. We use all pairs, i.e., UT-Zap50K-1, UT-Zap50K-2, and UT-Zap50K-lexi as training data. The pairwise comparisons in terms of *comfort* ($4,483$ pairwise comparisons), *open* ($4,351$) and *sporty* ($4,085$) attributes are taken as the user preferences for the training data, respectively. We exploit a pretrained ranker to augment more preferences for the training data. We pretrain a ranker to rank images in terms of the attribute from given pairwise comparisons and then construct pairwise preferences based on the scores evaluated by the ranker. If shoe image x_a is more *comfort*/*open*/*sporty* than shoe image x_b, then the preference is $s := x_a > x_b$.

Baselines: We consider GAN [Goodfellow et al., 2014], FBGAN [Gupta and Zou, 2019], GAN-RK [Saquil et al., 2018] (GAN plus an additional ranker) and GR (a generator with a ranker that is only trained with partial preferences) as our baselines. In terms of GAN, it is trained with three kinds of subsets: the entire data, the subset made up of local high-ranked samples (top-1 samples in the partial preferences), and the subset made up of global high-ranked samples (e.g., top 10% for MNIST; top 50% for LFW and UT-Zap50K) selected by a surrogate ranker. These three variants are denoted as GAN-0, GAN-1, and GAN-2, respectively.

Network & Hyperparameters: All methods are implemented based on WGAN-GP architecture (DCGAN version) [Gulrajani et al., 2017] unless specifically mentioned. For the training we use the Adam optimizer [Kingma and Ba, 2015] with learning rate $2*10^{-4}$ and $\beta_1 = 0.5, \beta_2 = 0.999$. According to Proposition 5 and Corollary 4, we set the ground-truth score vector for $s^{(R)}$ as $\boldsymbol{\pi}\left(s^{(R)}\right) = [10+5(l-1), 10+5(l-2), \ldots, 10, 0]$ for all datasets, which can make GARNet learn a user-desired data distribution. We simply set the ground-truth score vector for $s^{(G)}$ as $\boldsymbol{\pi}\left(s^{(G)}\right) = [10+5l, 10+5(l-1), \ldots, 10, 0]$. For MNIST, the batch size used is 50. The training iteration is set to $100K$. For LFW and UT-Zap50K, the batch size is 64. The training iteration is $200K$. The training images are resized to 32×32 unless specifically mentioned. To rank images in terms of certain attributes, we adopt a pairwise ranking loss in Burges et al. [2005]. The pretrained VGG16 [Simonyan and



(a) GAN-0 (b) GAN-1 (c) GAN-2 (d) FBGAN



(e) GAN-RK (f) GR (g) GARNet

Figure 5.6 Comparison w.r.t. the generation of user preferred digits (small digits) on MNIST.

Zisserman, 2015] is used to map an image to its ranking score. Our code develops based on Pytorch [Paszke et al., 2017].

Evaluation Metrics: (1) For MNIST, mean digit (MD) is adopted as a performance measure for the generation of preferred digits. The digit values for the generated data will be evaluated by a pretrained classifier for digit classification. For each method, we randomly generate $50K$ digits and calculate their MD.
(2) For LFW and UT-Zap50K, mean score (MS) is adopted as a performance measure for the generation of user preferred face images. The scores are evaluated by a pretrained ranker that learns to rank images in terms of the target attribute from the given preferences. Frechet Inception Distance (FID) [Heusel et al., 2017] measures the visual quality w.r.t. the high-ranked real face images. For each method, we randomly select $50K$ generated images and conduct the evaluation.

5.6.1 Preferences w.r.t. Discrete Digits

In this section, we apply GARNet to generate the user preferred digits on MNIST using the partial preferences.

According to Fig. 5.6 and Table 5.1, we highlight that only our GARNet successfully learns the distribution of the top-ranked digits from user preferences. Table 5.1 shows that except for GAN-0, GAN-RK, and GR, other methods have smaller MD compared to the training data (denoted as real). It means that those generative models can learn from user preferences to some extent. However, only FBGAN and GARNet can converge to generate

Table 5.1 Comparison of various methods on MNIST w.r.t. mean digit (MD, ↓). Best results are in bold. FBGAN suffers from mode collapse (Fig. 5.6d). GR generates meaningless results (Fig. 5.6f), so its MD is not collected.

Method	Real	GAN-0	GAN-1	GAN-2	GAN-RK	GR	FBGAN	GARNet
MD	4.45	4.49	0.75	0.15	4.63	-	**0.00**	**0.00**

top-ranked digits with MD 0.00. On the other hand, GAN-0 is the vanilla GAN that is trained with all training data, so its MD is close to that of real images. GAN-RK fails to learn from user preferences because of the conflict between the discriminator and ranker. GR cannot get any meaningful generations because it does not involve a distribution matching between real and generated samples. Therefore, its MD is not applicable.

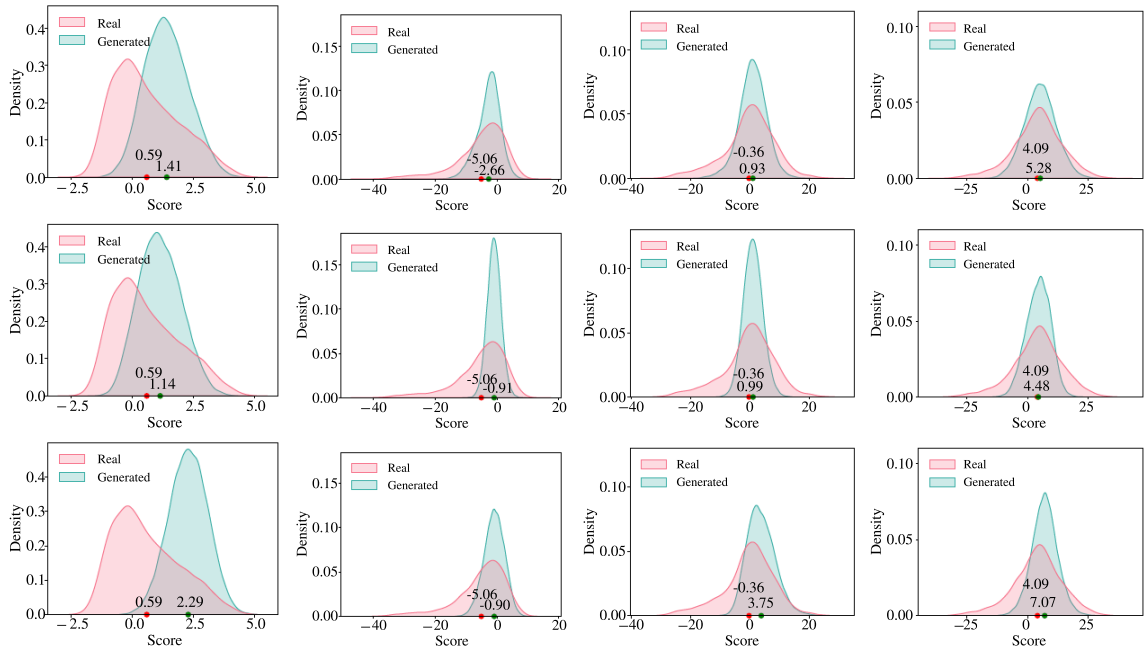
Fig. 5.6 shows that: (1) GARNet generates top-ranked digit (digit 0), with high quality as shown in Fig. 5.6g. (2) According to Fig. 5.6b, GAN-1 still generates digits that are not ranked top, like digit two, since the constructed training subset contains undesired samples. (3) GAN-2 achieves better performance than GAN-1 but still fails to converge to the distribution of the top-ranked digit. (4) FBGAN suffers from mode collapse of which the generated digit zeros are exactly the same. Meanwhile, it suffers from low-quality issues due to the introduction of generated samples.

5.6.2 Preferences w.r.t. a Continuous Attribute

As described in experiments on MNIST, GAN-1, GAN-2, FBGAN and our GARNet can learn from the user preferences. Therefore, we only compare GARNet with these baselines on generating user preferred facial images on LFW w.r.t. the *smile* attribute as well as generating user preferred shoe images on UT-Zap50K w.r.t. the *comfort*, *open* and *sporty* attribute, respectively.

Table 5.2 Comparison on LFW face data and UT-Zap50K shoe data w.r.t. performance measure (MS, ↑) and quality score (FID, ↓). The best results are highlighted in bold. Since FBGAN suffers from mode collapse (large FID; see its generated images in Appendix), its MS is not collected.

Dataset	MS				FID			
	Real	GAN-1	GAN-2	GARNet	GAN-1	GAN-2	FBGAN	GARNet
LFW (smile)	0.59	1.41	1.14	**2.29**	47.55	37.40	89.36	**22.22**
UT-Zap50K (comfort)	-5.06	-2.66	-0.91	**-0.90**	49.20	77.99	253.53	**39.16**
UT-Zap50K (open)	-0.36	0.93	0.99	**3.75**	63.92	112.93	265.49	**46.78**
UT-Zap50K (sporty)	4.09	5.28	4.48	**7.07**	41.02	84.59	431.44	**32.87**

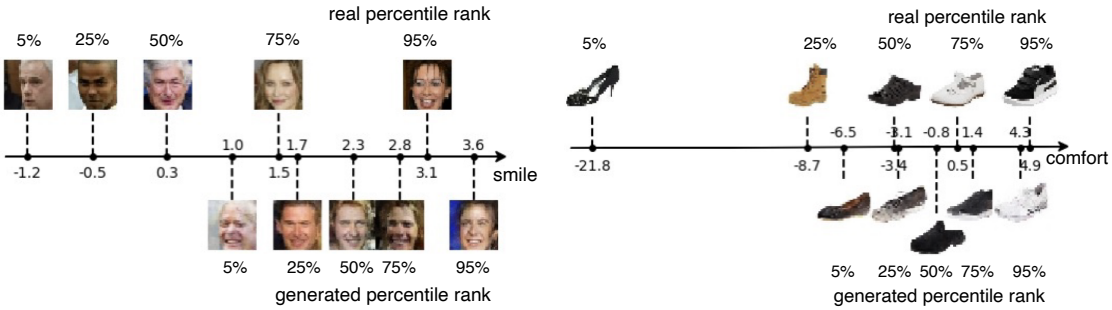


(a) *smile* (b) *comfort* (c) *open* (d) *sporty*

Figure 5.7 Comparison of GAN-1 (Row 1), GAN-2 (Row 2) and our GARNet (Row 3) in terms of score density. The green point denotes the mean score of generated samples while the red point denotes that of real samples.

We quantitatively evaluate the performance of learning from users' preferences by mean score (MS) and the generation quality by FID in Table 5.2. Meanwhile, we plot the score density for generated samples of various methods and training samples in Fig. 5.7. The evaluation results on LFW (*smile*) show that: GAN-1, GAN-2, and GARNet shift to a distribution of data with larger scores, which means they can learn from user preferences and generate face images with large *smiles*. (1) Our GARNet learns the best desired distribution, indicated by a best MS, along with best image quality, indicated by a lowest FID score. (2) GAN-1 and GAN-2 have poorer quality than GARNet. (3) FBGAN suffers from mode collapse and even generates meaningless images (see Appendix), which is verified by its large quality score (FID). Similar results can be concluded on UT-Zap50K.

For further visual quantification, we place GARNet's generated samples and training samples (a.k.a. real samples) w.r.t. their scores on the *smile* axis in Fig. 5.8a. GARNet mainly (over 95%) covers those real images ranked top 50% in terms of the *smile* attribute. GARNet even generates images with higher scores than the maximum scores in the LFW dataset, which shows its potential for generating data beyond existing rankings. We place GARNet's generated samples and training samples (a.k.a. real samples) w.r.t. their scores on the *comfort* axis in Fig. 5.8b. Over 75% generated images cover those real images ranked



(a) LFW (b) UT-Zap50K

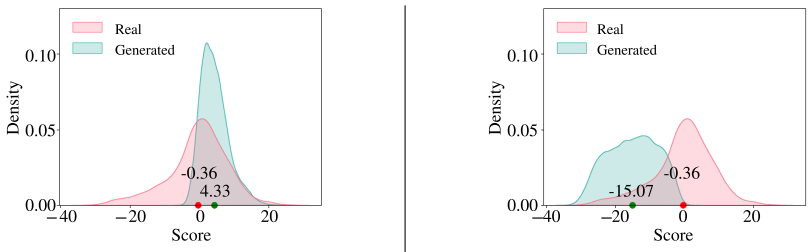
Figure 5.8 Real images (32×32, above the axis) vs. generated images (GARNet, below the axis) w.r.t. the score axis of attributes. The percentile rank of a given score is the percentage of scores in its frequency distribution that are less than that score. Real (generated) percentile rank means calculated among real (generated) images.

Figure 5.9 GARNet for multiple attributes. Images (64×64) are placed w.r.t. the "comfort" and "open" score. The results are obtained by a state-of-art GAN, thus have better quality.

top 50% in terms of the *comfort* attribute. Results on the *open* and *sporty* attributes can be seen in Appendix.

5.6.3 User Control on a Mixture of Preferences

To demonstrate that GARNet can generalize to a mixture of user preferences, i.e., preferences on multiple attributes, we conduct the experiments in terms of *comfort* and *open* attributes as the training data. To achieve a better generation, we apply the state-of-art GAN with a fixed augmentation probability 0.2 [Karras et al., 2020a]. Meanwhile, the training images are resized to 64×64 for better resolution.



(a) *open* (b) *not open*

Figure 5.10 The generated samples (32×32) and the score density (*open* ↑, score ↑) for CGARNet conditioned on *open* and *not open* attributes, respectively. The green point denotes the mean score of generated samples while the red point denotes that of real samples.

In Fig. 5.9, we visualize real images and generated images by ordering them using two pre-trained rankers defined over the *open* and *comfort* attributes, respectively. It is observed that GARNet generates shoe images that are more comfort or more open. In addition, the least *comfort* or the least *open* shoe images are not generated by GARNet as they are disliked by users.

It is worth mentioning that GARNet achieves a compromise between a pair of moderate conflicting attributes. Intuitively, *open* and *comfort* are conflicting attributes. For example, sport shoes are thought comfortable but may be slightly open. As shown in Figure 5.9, learning from preferences w.r.t. *open* and *comfort* attributes, GARNet converges to avoid generating samples ranked lowest in either of two attributes but generates relatively close and comfort shoes.

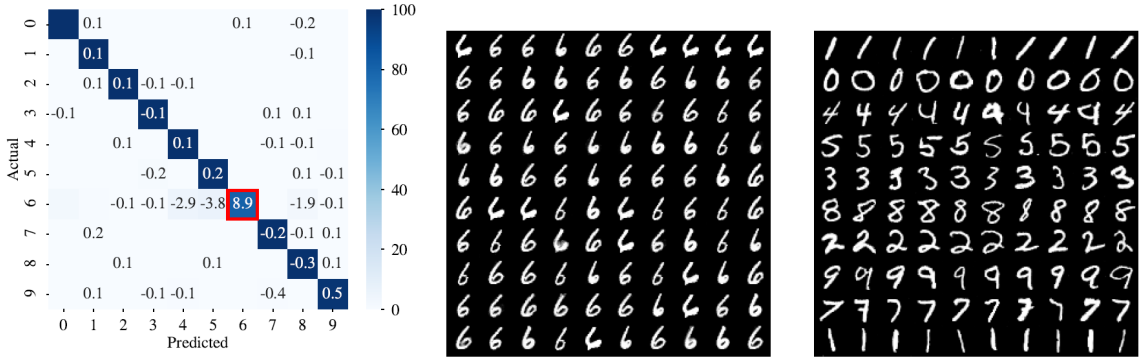
Conditional GARNet

We apply our conditional GARNet (CGARNet) for exact opposing attributes, i.e., *open* and *not open*.

Fig. 5.10 shows that: (1) When conditioned on *open* attribute, CGARNet generates shoe images with large *open* attribute values, with a score density that locates on a region of large *open* score values. (2) When conditioned on *not open* attribute, CGARNet generates shoe images with small *open* attribute values, with a score density that locates on a region of small *open* score values.

5.6.4 GARNet vs. GANs conditioned on Labels

In this section, we show that generation conditioned on preferences outperforms generation conditioned on labels in extremely imbalance class learning. Note conditional generation can



(a) Performance gain matrix (b) GARNet (c) Elastic-infoGAN

Figure 5.11 (a) GARNet boosts imbalance classification on MNIST with digit six as the minority class. The gain matrix (zero is not presented) is obtained by C'-C, where C' and C are the confusion matrix calculated on GARNet boosted data and original MNIST data, respectively. The color denotes the confusion matrix (%) on original MNIST data. (b-c) Visual results of GARNet and Elastic-infoGAN on MNIST extremely imbalanced data.

generate samples for the minority data, showing a promising application for imbalanced data classification by promoting the minority via data augmentation in a pre-processing manner.

Experimental setup: On the MNIST dataset [Lecun et al., 1998], we randomly pick up 0.5% samples of the digit six to constitute the minority class. All samples of the rest classes are retained. Note the imbalance ratio reaches to minor/major $\approx \frac{1}{200}$. We construct partial preferences simply by preferring digit six than any other class. We compare the classification performance of a CNN classifier (CNN) without and with data augmentation by GARNets.

Baselines: Elastic-infoGAN [Ojha et al., 2020] is a state-of-art generative model for imbalanced data generation. Standard GAN trained only with minor class is excluded as a baseline since there are insufficient training samples ($0.5\% * 4,951 \approx 25$) for training a generative model.

Fig. 5.11b shows that our GARNet successfully generates the user-desired data, i.e., digit six. Though there are limited digit six images in the training set, we can construct sufficient partial preferences between the minor class and the major classes. Therefore, our GARNet can learn a direction of user preferences and generate the desired data. In contrast, Fig. 5.11c shows that Elastic-infoGAN fails to generate digit six since it relies on class labels and the samples of digit six are too limited.

Therefore, our GARNet can be used to improve imbalance class learning via data augmentation but Elastic-infoGAN cannot. Fig. 5.11a shows the combined confusion matrix of two settings: with augmentation by GARNet and without augmentation. The color indicates the confusion matrix of CNN with imbalanced data, where every class has a high purity except digit six which is a bit confusing with digit four, five, and eight. By replenishing the

digit six with GARNets, we train the same CNN architecture (CNN-GARNet). It shows that the purity of digit six increases by 8.9%, demonstrating GARNets can efficiently augment the minority data in imbalanced scenarios.

5.7 Summary

This chapter presents a novel adversarial ranking framework, GARNet, to learn from human preferences. In particular, we prove that GARNet is equivalent to optimizing a divergence between the desired data distribution (determined by the given score vector) and the generated data distribution, which theoretically guarantees GARNet as a good estimator of the desired data distribution. Meanwhile, we empirically show GARNet can obtain corresponding distributions when different given score vectors are specified. Extensive experiments demonstrate GARNet can learn the distribution best matches user preferences compared to various baselines. A study of imbalanced class learning validates the advantage of preference-guided GARNet over GAN conditioned on labels.

The new generative modeling paradigm in the context of human preferences is especially proposed under the generative adversarial framework. It would be interesting to extend this paradigm that aligns the distribution of a user-specified subset of training data with the generative model to e.g. diffusion models [Ho et al., 2020].

Chapter 6

Refining Image-to-Image Translation by Rival Preferences

Having explored preference-guided desired data generation at the dataset level in Chapter 4 and Chapter 5, we target preference-guided desired data generation at the instance level in this chapter. We propose a new model TRIP for fine-grained image-to-image translation (namely, instance-level desired data generation) by using the preferences over image pairs on the strength of a specific attribute. In particular, we simultaneously train two modules: a generator that translates an input image to the desired image with smooth subtle changes with respect to the interested attributes; and a ranker that executes rival preferences consisting of the input image and the desired image. Rival preferences refer to the adversarial ranking process: (1) the ranker thinks no difference between the desired image and the input image in terms of the interested attributes; (2) the generator fools the ranker to believe that the desired image changes the attributes over the input image as expected. RAs over pairs of real images are introduced to guide the ranker to rank image pairs regarding the interested attributes only. With an effective ranker, the generator would "win" the adversarial game by producing high-quality images that present desired changes over the attributes compared to the input image. The experiments on two face image datasets and one shoe image dataset demonstrate that our TRIP achieves state-of-art results in generating high-fidelity images which exhibit smooth changes over the interested attributes.

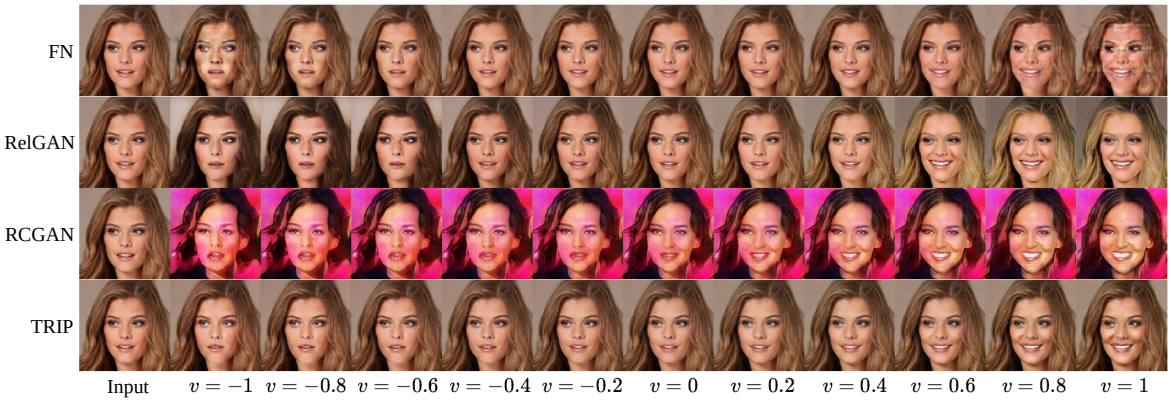


Figure 6.1 Fine-grained facial attribute ("smile") translation on CelebA-HQ dataset. v is a variable that controls the desired change of the "smile" attribute for the generated images.

6.1 Problem Statement

Let $\mathcal{X} = \{x_n\}_{n=1}^{N}$ denote a training dataset with N samples, $\mathcal{S} = \{s_m\}_{m=1}^{M}$ denote a collection of M preferences defined on subsets of $\mathcal{X}$. In particular, each $s \in \mathcal{S}$ is an ordered list, namely,

$$s = y \geq x,\ x, y \in \mathcal{X}. \tag{6.1}$$

Our target is to learn a generator that can generate a series of realistic versions of input image x that possess smooth changes on specific attributes by conditioning on a continuous attribute variable v, namely, $\hat{y} = G(x, v)$ (Fig. 6.1). Note that the preferences here refer to the preferences of pairs of images on the strength of the interested attributes, which is called "relative attributes" (RAs) in the remaining of this chapter.

Remark 7. *We consider pairs $\{(x, y)|y = x\}$, called "ties" in learning to rank for better ranking prediction [Zhou et al., 2008].*

6.2 TRIP for Fine-grained I2I translation

As clarified in Section 2.2.2, most existing work on I2I translation relies on binary attributes, obtaining unsatisfactory performance on fine-grained translation due to the limited description capacity of binary attributes. While some GAN-based works, i.e., RCGAN and RelGAN begun to improve fine-grained I2I translation by RAs, but they fail to reconcile the goal for fine-grained translation and the goal for high-quality generation, because their attribute model that is only trained with RAs on real images cannot generalize well to the generated images with interpolated RAs (See Section 2.2.2 for more details). In this section, we propose a new

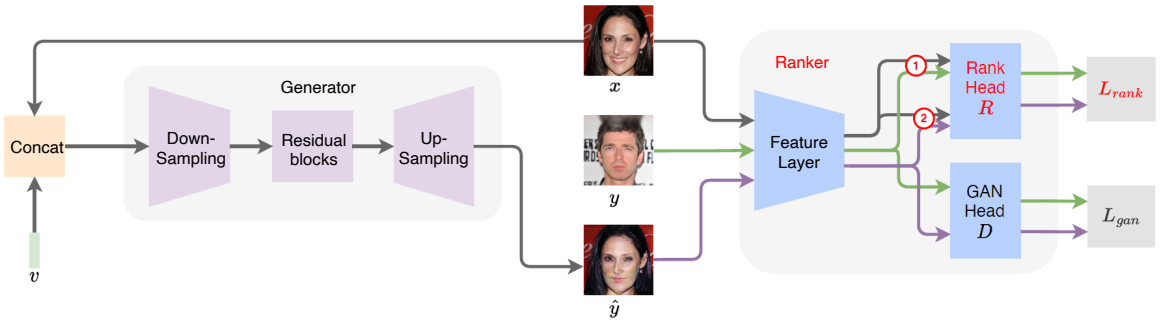


Figure 6.2 The network structure of TRIP. The main novelty is two folds: (1) the design of ranker and (2) the adversarial ranking process, which are denoted in red. ① and ② denote different image pairs, i.e., real image pairs and generated image pairs, corresponding to Fig. 6.3 and Fig. 6.4, respectively. R and D denote the rank head and the GAN head, respectively.

model, Translation via RIval Preferences (TRIP), feeds RAs on real images as well as on generated images to model RAs and maintains rival preferences to coordinate the two goals.

We open our description with the one-attribute case. The whole structure of TRIP is shown in Fig. 6.2, which consists of a generator and a ranker. The generator takes as input an image along with a continuous latent variable that controls the change of the attribute, and outputs the desired image; while the ranker delivers information in terms of image quality and the preference over the attribute to guide the learning of the generator. We implement the generator with a standard encoder-decoder architecture following [Wu et al., 2019b]. In the following, we focus on describing the detailed design of the ranker and the principle behind it. Then we propose our TRIP model and extend it to the multiple-attribute case. Last, we compare our TRIP with RCGAN and RelGAN in technical details.

6.2.1 Ranker for Relative Attributes

Relative attributes (RAs) are assumed to be most representative and most valid to describe the information related to the relative emphasis of the attribute, owing to its simplicity and easy construction [Parikh and Grauman, 2011, Saquil et al., 2018]. For a pair of images (x, y), RAs refer to their preference over the specific attribute: $y > x$ when y shows a greater strength than x on the target attribute and vice versa.

Pairwise learning to rank is a widely-adopted technique to model the relative attributes [Parikh and Grauman, 2011]. Given a pair of images (x, y) and its relative attribute, the pairwise

learning to rank technique is formulated as a binary classification [Cao et al., 2006], i.e.,

$$R(x,y) = \begin{cases} +1 & y > x; \\ -1 & y < x, \end{cases} \tag{6.2}$$

where $R(x,y)$ is the ranker's prediction for the pair of images (x,y) (Fig. 6.3).

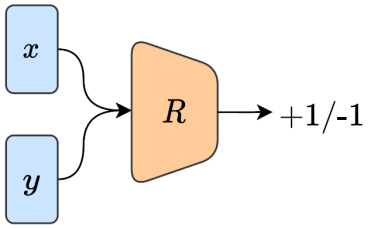


Figure 6.3 The ranker model to learn relative attributes for real image pairs.

.

Further, the attribute discrepancy between RAs, distilled by the ranker, can then be used to guide the generator to translate the input image into the desired one.

However, the ranker is trained on the real image pairs, which only focuses on the modeling of preference over the attribute and ignores image quality. To achieve the agreement with the ranker, the generator possibly produces unrealistic images, which conflicts with the goal of the discriminator.

6.2.2 Ranking Generalization by Rival Preferences

According to the above analysis, we consider incorporating the generated image pairs into the modeling of RAs, along with the real image pairs, to reconcile the goal of the ranker and the discriminator. Consequently, the resultant ranker will not only generalize well to the generated pairs but also avoid providing untrustworthy feedback by distinguishing the unrealistic pairs from the real pairs.

Motivated by the adversarial training of GAN [Goodfellow et al., 2014], we introduce an adversarial ranking process between a ranker and a generator to incorporate the generated pairs into the training of ranker. To be specific,

- *Ranker.* Inspired by semi-supervised GAN [Odena, 2016], we assign a pseudo label to the generated pairs. In order to avoid a biased influence on the ranking decision over real image pairs, i.e., positive (+1) or negative (-1), the pseudo label is designed to be zero. Note that a generated pair consists of a synthetic image and its input in order to connect

the ranker prediction to the latent variable.

$$R(x,\Delta) = \begin{cases} +1 & (\Delta = y) \wedge (y > x); \\ -1 & (\Delta = y) \wedge (y < x); \\ 0 & \Delta = \hat{y}. \end{cases} \tag{6.3}$$

where $\hat{y}$ denotes the output of the generator given the input x and an attribute variable v, i.e., $\hat{y} = G(x,v)$. In particular, v controls the desired change of attributes for the generated images. Δ is a placeholder that can be either a real image y or a generated image $\hat{y}$.

- *Generator.* The goal of the generator is to achieve the consistency between the ranking prediction $R(x,\hat{y})$ and the corresponding latent variable v. When $v > 0$, the ranker is supposed to believe that the generated image $\hat{y}$ has a larger strength of the specific attribute than the input x, i.e., $R(x,\hat{y}) = +1$; and vice versa.

$$R(x,\hat{y}) = \begin{cases} +1 & v > 0; \\ -1 & v < 0. \end{cases} \tag{6.4}$$

We denominate the opposite goals between the ranker and the generator w.r.t. the generated pairs as *rival preferences*[1]. Such preferences extend the adversarial game defined on binary classification [Goodfellow et al., 2014] to ranking. An intuitive example of the rival preferences is given in Fig. 6.4 for better understanding.

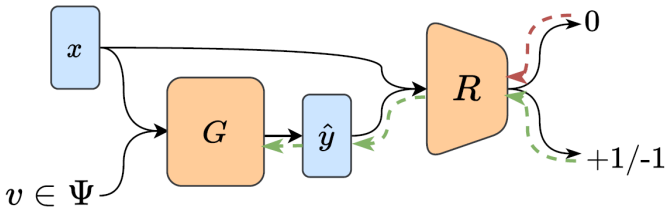


Figure 6.4 Rival Preferences for the generated image pairs between the ranker and the generator, which ensure the ranker generalizes well to the generated pairs. Ψ denotes $[-1,0) \cup (0,1]$.

The ranker is promoted in terms of the following aspects: (1) The function of the ranker on the real image pairs is not changed. First, the ranker learns to correctly rank real image pairs by optimizing the first term in Eq. (6.6a). Second, the generated image pairs are uniformly sampled regarding their latent variables $v \in [-1,1]$, which contain positive

[1] "Rival" is not only used to denote "adversaral", but also more in compliance with the theme of ranking since it has an extra meaning as "competitive".

rankings ($\text{sgn}(v) = +1$) when conditioned on $v > 0$ and negative rankings ($\text{sgn}(v) = -1$) when conditioned on $v < 0$. By assigning label "0" (the second term in Eq. (6.6a)), the ranking output for all generated image pairs can thus be neutralized while ensuring the ranking performance on the real image pairs. (2) The ranker avoids providing biased ranking prediction for unrealistic image pairs. As we constrain the generated pairs at the decision boundary, i.e, $R(x, \hat{y}) = 0$, the ranker is invariant against large variances of generated pairs [Chapelle et al., 2008]. Especially, the influence of unrealistic features contained in the generated pairs on the ranking decision is suppressed. (3) The ranker can capture the exclusive difference over the specific attribute through the adversarial process. Since the ranker rejects to give effective feedback for unrealistic image pairs, only the realistic image pairs can attract the attention of the ranker. Therefore, the ranker only passes the effective information related to the target attribute to the generator.

The rank head loss (Eq. (6.6)) is effective for smooth translation, which will be empirically validated in Section 6.4.4. Nevertheless, the ranking loss focuses on the difference of a pair of samples, so it is somehow inferior for the relativistic quality of single samples. We empirically find that our adversarial ranking loss can guarantee a superior attribute-specified quality but only boost a relatively good realistic quality (Section 6.4.4). To further improve the realistic quality, we introduce a parallel GAN head following the feature layer to ensure the image quality together with a rank head, shown in Fig. 6.2. The GAN head loss focuses on the quality of single samples and can further improve the ranker's ability to distill realistic features. The function of different losses is summarized in Table 6.1.

According to the above analysis, the ranker would not evoke conflicts with the goal of the image quality. Therefore, we successfully reconcile the two goals of image quality and the extraction of the attribute difference. With a powerful ranker, the generator would "win" the adversarial game by producing realistic pairs consistent with the latent variable.

Remark 8 (Assigning label 0 to similar real image pairs)**.** *We follow Parikh and Grauman [2011] to assign label 0 to pairs $\{(x,y)|y = x\}$, called "ties" in learning to rank [Zhou et al., 2008]. Label 0 denotes that there is no difference between x and y w.r.t. the specified attribute. The learning of these pairs can further improve the ranking prediction [Zhou et al., 2008]. Note that assigning label 0 to the similar real pairs and the generated pairs have a same physical meaning.*

6.2.3 Linearizing Ranking Output for Smooth Translation

Eq. (6.4) models the relative attributes of the generated pairs as a binary classification, which would fail to enable a fine-grained translation since the subtle changes implied by the latent variable are not distinguished by the ranker. For example, given $v_1 > v_2 > 0$, the ranker gives same feedbacks, i.e., $\text{sgn}(v_1) = \text{sgn}(v_2) = +1$, for $(x, \hat{y}_1)$ and $(x, \hat{y}_2)$, where $\hat{y}_* = G(x, v_*)$. This loses the discrimination between the two pairs. To achieve the fine-grained translation, we linearize the ranker's output for the generated pairs so as to align the ranker prediction with the latent variable. We thus reformulate the binary classification as the regression:

$$R(x, \hat{y}) = v. \tag{6.5}$$

Given two latent variables $1 > v_2 > v_1 > 0$, the ranking prediction for the pair generated from v_2 should be larger than that from v_1, i.e., $1 > R(x, \hat{y}_2) > R(x, \hat{y}_1) > 0$. The ranker's outputs for the generated pairs would be linearly correlated to the corresponding latent variable. Since the ranker's output for a generated pair reflects the difference between the generated image and the input image, the generated image can change smoothly over the input image according to the latent variable.

As explained in Section 6.2.2, the ranker's generalization to the generated pairs is boosted by rival preferences. On the other hand, to make the ranker better generalize to have linearized ranking outputs for the generated pairs, we tailor-design the structure for the ranker and adopt the least square loss for the ranking predictions. Specifically, (1) the rank head in the ranker is designed to output real values $R(x, y)$, i.e., the last layer being the linear dense layer (Table 1 in Supplementary). Denoting the mapping before the last layer as f and the parameter of the last layer as w, the term $(R(x, y) - r)^2$ in Eq. (6.6a) can be reformulated as $(w^T f(x, y) - r)^2$. This actually learns the prediction of discrete RAs as a regression problem, which makes it easy to generalize the ranker to have a continuous ranking predictions. (2) The least square loss allows our ranker to produce soft prediction values for real discrete RAs since the loss has a small penalty when the prediction is close to the ground truth (Fig. 2 [2] in [Martins and Astudillo, 2016]). Such flexibility enables our ranker to capture the discrepancy of different real image pairs w.r.t. the specific attribute. As a result, our ranker is capable to provide a continuous ranking predictions [3]

[2] Comparison among commonly used losses (including the least square loss and the cross-entropy loss) for binary classification.

[3] In Fig. 6.15, the ranker's outputs for the real image pairs locate at a neighborhood area of -1, 0, and 1 when convergence (iteration 99K), which almost covers the range from -1 to 1.).

Table 6.1 The function of different losses in our TRIP.

Loss	Smooth translation	High-quality generation	
		Realistic	Attribute-specified
Rank head (Eq. (6.6))	✓	✓	✓
GAN head (Eq. (6.7))	✗	✓	✗

6.2.4 Translation via RIval Preferences (TRIP)

In the following, we introduce the loss functions for the two parallel heads in the ranker. The overall network structure can be seen in Fig. 6.2.

Loss of rank head R: the loss function for the rank head and the generator is defined as:

$$L_{rank}^{R} = \mathbb{E}_{p(x,y,r)}\left[\left(R\left(x,y\right)-r\right)^2\right] + \lambda \mathbb{E}_{p(x)p(v)}\left[\left(R\left(x,G(x,v)\right)-0\right)^2\right]; \tag{6.6a}$$

$$L_{rank}^{G} = \mathbb{E}_{p(x)p(v)}\left[\left(R\left(x,G(x,v)\right)-v\right)^2\right], \tag{6.6b}$$

where $r = \begin{cases} 1 & y > x \\ 0 & y = x \\ -1 & y < x \end{cases}$ denotes the relative attribute and v is the latent variable to control the attribute changes. $p(x,y,r)$ are the joint distribution of real image preferences. Let $\hat{y} = G(x,v)$. $p(x)$ is the distribution of the training images. $p(v)$ is a uniform distribution on $[-1,1]$. λ is the weight factor that determines the strength of adversarial training between the ranker and the generator.

By optimizing L_{rank}^{R} (Eq. (6.6a)), the ranker is trained to predict correct labels for real image pairs and assign label 0 for generated pairs, i.e., Eq. (6.3). By optimizing L_{rank}^{G} (Eq. (6.6b)), the generator is trained to output the desired image $\hat{y}$, where the difference between $\hat{y}$ and x is consistent with the latent variable v, i.e., Eq. (6.5). Therefore, the ranker learns to distill the discrepancy from the interested RAs. Meanwhile, the two rival goals on the generated pairs raise an adversarial training between the ranker and the generator. That is, the generator urges the ranker to predict that there are desired changes in the generated images while the ranker predicts that there are no changes. The competitive game promotes the improvement of two modules simultaneously.

Loss of GAN head D: to be consistent with the above rank head and also ensure a stable training [4], a regular least square GAN's loss is adopted:

$$L_{gan}^{D} = \mathbb{E}_{p(x)}\left[\left(D\left(x\right)-1\right)^{2}\right] + \mathbb{E}_{p(x)p(v)}\left[\left(D\left(G(x,v)\right)-0\right)^{2}\right]; \tag{6.7a}$$

$$L_{gan}^{G} = \mathbb{E}_{p(x)p(v)}\left[\left(D\left(G(x,v)\right)-1\right)^{2}\right], \tag{6.7b}$$

where 1 denotes the real image label while 0 denotes the fake image label.

Jointly training the rank head and the GAN head, the gradients backpropagate through the shared feature layer to the generator. Then our TRIP can conduct the high-quality fine-grained I2I translation.

6.2.5 Extended to the Multiple Attributes

To generalize TRIP to multiple (K) attributes, we use vectors $\vec{v}$ and $\vec{r}$ with K dimension to denote the latent variable and the preference label, respectively. Each dimension controls the change of one interested attribute. In particular, the ranker consists of one GAN head and K parallel rank head. The overall loss function is summarized as follows:

$$L_{rank}^{R} = \mathbb{E}_{p(x,y,\vec{r})}\sum_{k}\left[\left(R_{k}\left(x,y\right)-\vec{r}_{k}\right)^{2}\right]] + \lambda\mathbb{E}_{p(x)p(\vec{v})}\sum_{k}\left[\left(R_{k}\left(x,G(x,\vec{v})\right)-0\right)^{2}\right]; \tag{6.8a}$$

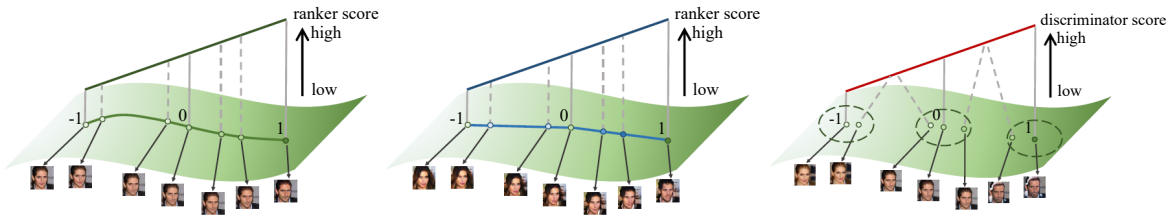
$$L_{rank}^{G} = \mathbb{E}_{p(x)p(\vec{v})}\sum_{k}\left[\left(R_{k}\left(x,G(x,\vec{v})\right)-\vec{v}_{k}\right)^{2}\right], \tag{6.8b}$$

where R_k is the output of the k-th rank head. $\vec{v}_k$ and $\vec{r}_k$ are the k-th dimension of $\vec{v}$ and $\vec{r}$, respectively.

6.3 Discussions

In this section, we analyze the advantages of TRIP in technical details. For better understanding, we compare TRIP with the two most related state-of-art works, i.e., RCGAN and RelGAN. In particular, we plot the correspondence between the generated pairs (consisting of the generated images and the coupled input images) and the ranker score in Fig. 6.5. For simplicity of presentation, we omit the input images.

[4]The least square GAN loss is demonstrated to possess superior training stability compared to the original GAN loss in [Mao et al., 2017].



(a) TRIP (b) RCGAN (c) RelGAN

Figure 6.5 Correspondence between the ranker and the generated images by TRIP, RCGAN, and RelGAN, respectively. *TRIP:* the generated images (deep green line) lie on the data manifold (green curved surface), which have high quality. Meanwhile, they are linearly correlated to the ranker scores, which delivers smooth translation. *RCGAN:* the generated images (blue line, as distinct from the color of data manifold) are out of the data manifold, although exhibiting linear correlation with the ranker output. *RelGAN:* the generated images gather on the data main manifold within three green circles, and fail to spread out linearly with its discriminator score.

6.3.1 TRIP Compared with RCGAN

Different critics (to guide the generator). Our TRIP applies an adversarial ranker [5] (Fig. 6.2), to reconcile the goal for fine-grained translation and the goal for high-quality generation (See Fig. 6.5a.). RCGAN applies two critics to model the two goals separately: a vanilla ranker [6] to model the goal for fine-grained translation and a discriminator to model the goal for high-quality generation. However, the two goals are difficult to coordinate during the training stage.

Adversarial ranking vs. Vanilla ranking (namely, whether generated pairs are used to train the ranker or not). Except real image pairs, the ranker of our TRIP is also trained with the generated pairs, which targets an opposite goal of the generator. With such adversarial ranking, our ranker can have better generalization to the generated pairs (explained in Section 6.2.2). By contrast, the ranker of RCGAN is only trained with real image pairs (vanilla ranking), which generalizes poorly to the generated image pairs and would induce the generator to generate out-of-data manifold images (empirically verified in Section 6.4.1; see Fig. 6.5b.).

Regression-based ranking loss vs. Binary classification-based ranking loss. Benefiting from the least square loss, we can easily generalize the ranker to have continuous ranking predictions on the generated pairs to guide the generator for smooth translation (See the explanation in Section 6.2.3.). The ranker of RCGAN uses binary classification (sigmoid)

[5] A ranker equipped with adversarial ranking.
[6] A ranker equipped with vanilla ranking.

based ranking loss on the generated pairs to guide the generator, which is inferior for smooth translation (empirically verified in Section 6.4.1 and Section 6.4.3).

6.3.2 TRIP Compared with RelGAN

Ranker vs. Matching-aware discriminator. TRIP distills the attribute difference by modeling RAs with a ranking model. In contrast, RelGAN resorts to a matching-aware discriminator to learn attribute information. Specifically, the discriminator judges whether an input-output pair matches the RAs or not with binary classification. Namely, the pair (x, y) along with its ground-truth RA label r is classified as 1 while the pair (x, y) along with the wrong RA label r is classified as 0. Compared to a binary classifier, the ranker in TRIP is more effective to capture subtle differences exclusively regarding the target attributes.

Adversarial ranking vs. Interpolation. TRIP introduces an adversarial ranking process, ensuring that the ranker can criticize the generated images with continuous RAs during its training process. Thus, TRIP can perform fine-grained control over the (ranker-guided) generator regarding the target attributes. Instead, RelGAN generalizes to output images with continuous attribute differences by resorting to interpolation over the discrete RAs, which is ill-defined [Berthelot et al., 2018] and needs extra constraints (interpolation loss, which will be discussed in the following point).

Maintaining the linear tendency or not. TRIP: recalling Eq. (6.5), the ranker's outputs for the generated pairs are linearly correlated to the corresponding latent variables (See Fig. 6.5a). So the generated image $G(x, v)$ can change smoothly over the input image x according to the RA. RelGAN: recalling its interpolation loss in Sect 3.5 of [Wu et al., 2019b],

$$\min_G \mathbb{E}_{p(x)p(r)p(\alpha)}[\|D_{\text{Interp}}(G(x, \alpha r))\|^2],$$

where $r \in \{-1, 0, 1\}$ and $\alpha \in [0, 1]$. D_{Interp} recovers the interpolation ratio and assigns label 0 to non-interpolated images. Such interpolation loss aims at forcing interpolated images to be indistinguishable from non-interpolated images, which is designed for high-quality interpolation but destroys smooth translation (See Fig. 6.5c). Experiment results in Section 6.4.1 and Section 6.4.3 verify our claim.

6.4 Experiments

In this section, we compare our TRIP with various baselines on the task of fine-grained I2I translation. We then verify that our ranker can distinguish the subtle difference in a pair of images. Thus, we propose to apply our ranker for evaluating the fine-graininess of image

pairs generated by various methods. We finally extend TRIP to the translation of multiple attributes.

Datasets. All experiments are conducted in the real-world datasets with discrete RAs $r \in \{+1, 0, -1\}$. In particular, we use two face image datasets, i.e., the high quality subset of Celeb Faces Attributes Dataset (CelebA-HQ) [Karras et al., 2018] and Labeled Faces in the Wild with attributes (LFWA) [Liu et al., 2015], and one shoe image dataset, i.e., UT-Zappos50K dataset (UT-Zap50K) [Yu and Grauman, 2014]. CelebA-HQ consists of 30K face images of celebrities with 40 binary attributes. LFWA has $13,143$ images with 73 binary attributes. UT-Zap50K contains $50,025$ catalog shoe images together with corresponding edge images [Isola et al., 2017], where the binary label is 1 when an image is a shoe image and is 0 when an image is an edge image. We resize the images to 256×256 for three datasets. The relative attributes are obtained for any two images x and y based on the binary labels, following the former work [Saquil et al., 2018, Wu et al., 2019b]. Thus we can make a fair comparison with other baselines in terms of same supervision information. For instance, for "smile" attribute, we construct the comparison $x > y$ when the "smile" label of x is 1 while the "smile" label of y is 0, and vice versa. We find that all methods fail to achieve satisfactory results on the shoe dataset using the discrete relative attributes. To enhance the fine-grained translation on UT-Zap50K, we augment the training images by mixing up shoe images with the corresponding edge images [Zhang et al., 2018]. Specifically, a mix-up image is obtained by $(1-\gamma)x+\gamma y$, where y is a shoe image and x is y's corresponding edge image. The corresponding mix-up label is γ, where $\gamma \in (0,1)$.

Implementation Details. As the translation is conducted on the unpaired setting, the cycle consistency loss L_{cycle} is usually introduced to keep the identity of faces when translation [Zhu et al., 2017, Wu et al., 2019b, He et al., 2019]. An orthogonal loss L_o and the gradient penalty loss L_{gp} are added to stabilize the training following [Wu et al., 2019b]. The weighting factors for L_{gan}, L_{cycle}, L_o and L_{gp} are λ_g, λ_c, λ_o and λ_{gp}, respectively. Except $\lambda_g = 0.5$ for CelebA-HQ, $\lambda_g = 5$ for LFWA and $\lambda_g = 2.5$ for LFWA, we set the same parameter for all datasets. Specifically, we set $\lambda = 0.5$, $\lambda_c = 2.5, \lambda_{gp} = 150, \lambda_o = 10^{-6}$. We use the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to $1e$-5 for the ranker and $5e$-5 for the generator. The batch size is set to 4. We split the dataset into training/test with a ratio $90/10$ for face datasets. We use 40K shoe-edge image pairs (80K images in total) and the corresponding mix-up images (40K images in total) as training data and $10,025$ shoe-edge image pairs ($20,050$ images in total) as test data for UT-ZAP50K dataset. We pretrain our TRIP only with L_{gan} to enable a good reconstruction for the generator. By doing so, we ease the training by sequencing the learning of our TRIP. That is, we first make a generation with good quality. Then when L_{rank} begins to be used to train TRIP, our ranker can mainly

focus on the relationship between the generated pairs and its corresponding conditional v, rather than handling the translation quality and the generation quality simultaneously. All the experiment results are obtained by a single run. Each run contains 100K iterations.

Baselines. We compare TRIP with FN [Lample et al., 2017], RelGAN [Wu et al., 2019b] and RCGAN [Saquil et al., 2018]. We use the released codes of FN, RelGAN and RCGAN.

Evaluation Metrics. Following [Wu et al., 2019b], we use four metrics to quantitatively evaluate the performance of fine-grained translation. Standard Deviation of Structural SIMilarity (DSSIM) measures the performance of fine-grained translation. Mean Square Error (MSE) and Frechet Inception Distance (FID) measure the visual quality for shoe dataset and face datasets, respectively. Accuracy of Attribute Swapping (AAS) evaluates the accuracy of the binary image translation. The swapping for the attribute is to translate an image, e.g., from "smile" to "no smile".

- *DSSIM.* We first apply the generator to produce a set of fine-grained output images $\{x_1, \ldots, x_l\}$ by conditioning an input image and a set of latent variable values from -1 to 1 with a step 0.2. Then $l = \frac{1-(-1)+0.2}{0.2} = 11$. We compute the standard deviation of Structural SIMilarity (SSIM) between x_{i-1} and x_i as follows:

$$\text{DSSIM} = \sigma\left(\{\text{SSIM}\left(x_{i-1}, x_i\right) \mid i = 1, \cdots, 11\}\right). \tag{6.9}$$

  We calculate DSSIM for each image from the test dataset and average them to get the final score. Smaller DSSIM denotes better smooth translation.

- *MSE.* For shoe→edge in UT-ZAP50K, the input-output examples are paired. Namely, the input image (shoe) and the output image (edge) only differ in their style, i.e., photo style and edge style, respectively. Then the mean square error between the translated images from generative models and the ground-truth output images can be calculated to measure the visual quality of translation. We conduct translation for all images in UT-ZAP50K dataset, obtaining $50,025$ generated pairs, and calculate the MSE. Smaller MSE denotes generation with better quality.

- *FID.* As the input-ouptut examples in face datasets are unpaired, we evaluate the statistical difference between the translated images and the training dataset to measure the visual quality. It is calculated with $30K$ translated images on CelebA-HQ dataset and $13,143$ translated images on LFWA dataset. Smaller FID denotes generation with better quality.

- *AAS.* The accuracy is evaluated by a facial attribute classifier that uses the Resnet-18 architecture [He et al., 2016]. To obtain AAS, we first translate the test images with the trained GANs and then apply the classifier to evaluate the classification accuracy of the

Figure 6.6 Comparison of fine-grained facial attribute ("smile") translation on LFWA dataset.

translated images coupled with its swapping attribute. Higher accuracy means that more images are translated as desired.

Figure 6.7 Comparison of fine-grained translation ("shoe→edge") on UT-Zap50K dataset.

Table 6.2 Fine-grained performance (DSSIM) and image quality (FID/MSE) of FN, RCGAN, RelGAN and TRIP on CelebA-HQ, LFWA and UT-ZAP50K. The best results are highlighted in bold. Considering the value range, we round four decimal places for DSSIM and round two decimal places for FID and MSE. RCGAN fails to make fine-grained translations w.r.t. the "mouth" attribute on CelebA-HQ and "shoe→edge" on UT-ZAP50K. So we do not collect their results.

Model	Fine-grained (DSSIM)							Image quality (FID/MSE)						
	CelebA-HQ				LFWA		UT-Zap50K	CelebA-HQ (FID)				LFWA (FID)		UT-Zap50K (MSE)
	Smile	Gender	Mouth	Cheekbones	Smile	Frown	shoe→edge	Smile	Gender	Mouth	Cheekbones	Smile	Frown	shoe→edge
FN	0.0122	**0.0036**	0.0075	0.0039	0.0066	0.0049	0.0395	41.48	48.66	42.79	43.15	**12.59**	**11.37**	6987.76
RCGAN	0.0084	0.0138	-	0.0099	0.0079	0.0106	-	398.05	418.06	-	385.27	437.93	425.59	-
RelGAN	0.0512	0.0924	0.0261	0.0510	0.0137	0.0159	0.0023	10.92	31.29	9.55	10.28	16.76	16.21	6973.37
TRIP	**0.0030**	0.0077	**0.0017**	**0.0028**	**0.0008**	**0.0005**	**0.0010**	**10.19**	**26.47**	**7.18**	**9.50**	22.25	23.65	**6187.88**

Figure 6.8 Image comparison of TRIP (second row) with RelGAN (first row) w.r.t. the "smile" attribute. The 1st column is input image. Other columns are generated images conditioning on $v = -1, -0.5, 0.5, 1$ from left to right.

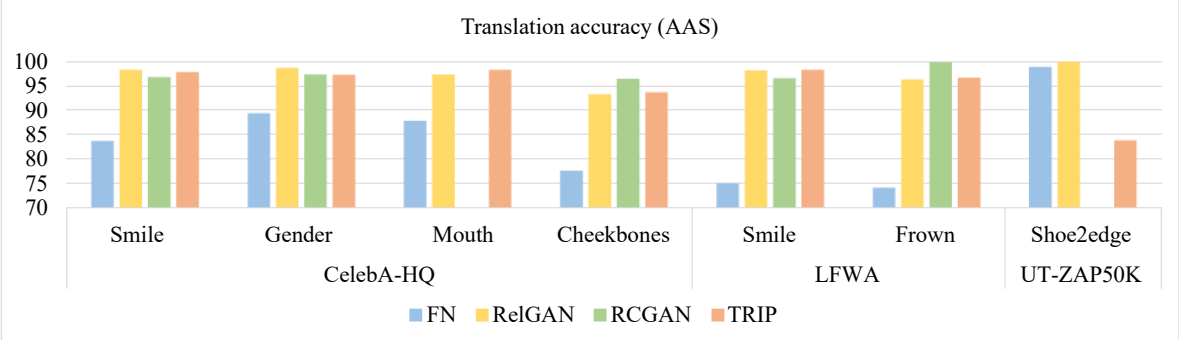


Figure 6.9 Translation accuracy (AAS, higher is better) of FN, RCGAN, RelGAN and TRIP on CelebA-HQ, LFWA and UT-ZAP50K. RCGAN fails to make fine-grained translations w.r.t. the "mouth" attribute on CelebA-HQ and "shoe→edge" on UT-ZAP50K. So we do not collect their results.

6.4.1 Fine-grained Image-to-Image Translation

We conduct fine-grained I2I translation on a single attribute. On CelebA-HQ dataset, we translate images in terms of "smile", "gender", "mouth open" and "high cheekbones" attributes, respectively. On LFWA dataset, we translate images in terms of "smile" and "Frown" attributes, respectively. On UT-ZAP50K dataset, we translate the shoe images to edge images, i.e., "shoe→edge". We show that our TRIP achieves the best performance on the fine-grained I2I translation task comparing with various strong baselines in the following three metrics.

Visual results. Fig. 6.1, Fig. 6.8 and Fig. 6.6 show that: (1) all methods can translate the input image into "more smiles" when $v > 0$ or "less smiles" when $v < 0$ on CelebA-HQ and LFWA, respectively. The degree of changes is consistent with the numerical value of v. (2) Our TRIP's generation achieves the best visual quality, generating realistic output images

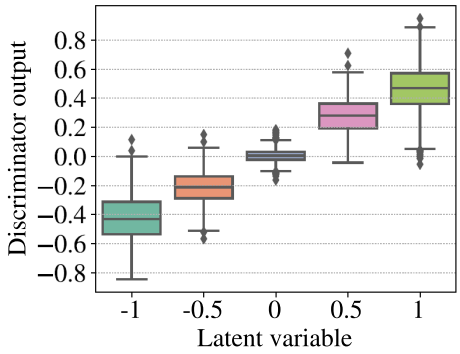


Figure 6.10 The box plot of the ranker's output for generated pairs with different values of the latent variable.

that are different from the input images only in the specific attribute. In contrast, FN suffers from image distortion issues. RelGAN's generation not only changes the specific attribute "smile", but is also influenced by other irrelevant attributes, e.g., "hair color". RCGAN exhibits extremely poor generation results. Fig. 6.7 shows that: FN and TRIP can translate shoe images into edge images smoothly but RelGAN fails.

Fine-grained score. We present the quantitative evaluation of the fine-grained translation in Table 6.2. Our TRIP achieves the lowest DSSIM scores for three datasets, consistent with the visual results. Note that a trivial case to obtain a low DSSIM is when the translation is failed. Namely, the generator would output the same image no matter what the latent variable is. Therefore, we further apply AAS to evaluate the I2I translation in a binary manner. RelGAN, RCGAN and TRIP mostly achieve over 95% accuracy except for FN (Fig. 6.9). Under this condition, it guarantees that a low DSSIM indeed indicates the output images change smoothly with the latent variable.

Image quality score. Table 6.2 presents the quantitative evaluation of the image quality. (1) Our TRIP achieves the best image quality with the lowest FID scores. (2) FN achieves the best FID on LFWA dataset. Because the FN achieves a relatively low accuracy of the translation, $< 75\%$ in Fig. 6.9, many generated images would be the same as the input image. It means that the statistics of the translated images are similar to that of the input images, leading to a low FID. (3) RCGAN has the worst FID scores, consistent with the visual results in Fig. 6.1 and Fig. 6.6.

6.4.2 Physical Meaning of Ranker Output

As shown in Fig. 6.10, (1) for a large v, the ranker would output a large prediction. It demonstrates that the ranker indeed generalizes to synthetic imaged pairs and can discriminate the subtle change among each image pair. (2) The ranker can capture the whole ordering

instead of the exact value w.r.t. the latent variable. Because the ranker that assigns 0 to the generated pairs inhibits the generator's loss optimizing to zero, although our generator's objective is to ensure the ranker output values are consistent with the latent variable. However, the adversarial training would help the ranker to achieve an equilibrium with the generator when convergence, so that the ranker can maintain the whole ordering regarding the latent variable.

From Fig. 6.1 and Table 6.2, it shows that when conditioning on different latent variables, our TRIP can translate an input image into a series of output images that exhibit the corresponding changes over the attribute. We then evaluate the function of our ranker using these fine-grained generated pairs. It verifies that our ranker's output well-aligns to the relative change in the pair of images.

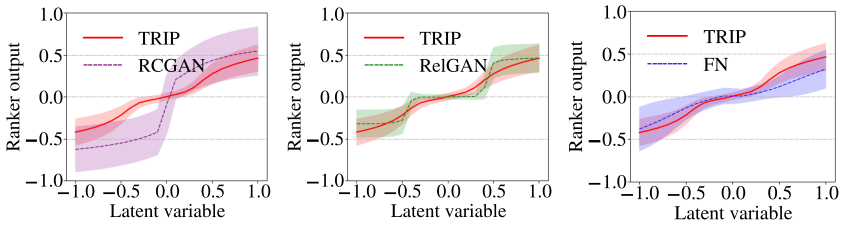
We further evaluate fine-grained I2I translations w.r.t. the "smile" attribute on the test dataset of CelebA-HQ (Fig. 6.10). The trained generator is applied to generate a set of $G(x,v)$ by taking as inputs an image x and $v = -1.0, -0.5, 0.0, 0.5, 1.0$, respectively. Note we use the test images with smiles for a negative v, or the test samples without smiles otherwise. Then we collect the output of the ranker for each generated pair and plot the density in terms of different v.

6.4.3 Linear Tendency on the Latent Variable

As our ranker can reveal the changes between image pairs, which is verified in the section 6.4.2, we use it to evaluate the subtle differences between the fine-grained synthetic image pairs generated by various baselines.

We generate the fine-grained pairs on the test dataset of CelebA-HQ w.r.t. the "smile" attribute. Each trained model produces a series of synthetic images by taking as input a real image and different latent variables. The range of the latent variable is from -1 to 1 with step 0.1. Then the ranker, pre-trained by our TRIP, is applied to evaluate the generated pairs and group them in terms of different conditioned latent variables for different models, respectively. In terms of each group, we calculate the mean and the standard deviation (std) for the outputs of the ranker (Fig. 6.11).

Fig. 6.11 shows that (1) the ranking output of TRIP exhibits a linear trend with the lowest variance w.r.t. the latent variable. This demonstrates that TRIP can smoothly translate the input image to the desired image over the specific attribute along the latent variable. (2) The ranking output of RCGAN behaves like a tanh curve with a sudden change when the latent variable is around zero. It means that RCGAN cannot smoothly control the attribute strength for the input image. In addition, RCGAN has the largest variance on the ranking output due to the low quality of the generated images, which introduces noises to the ranker's



v	RelGAN	RCGAN	FN	TRIP
−1.	0.165	0.268	0.256	**0.157**
−0.5	0.164	0.265	0.137	**0.107**
0.	**0.030**	0.354	0.089	0.042
0.5	0.183	0.273	0.122	**0.121**
1.	0.172	0.290	0.222	**0.159**

Figure 6.11 The first three subfigures plot the ranker's output for generated pairs in terms of different latent variables. The curve shows the mean of the output, while the shaded region depicts the standard deviation of the output. We summarize the standard deviation in the table for better understanding.

Figure 6.12 Qualitative evaluation of ablation study on CelebA-HQ ("smile" attribute). L_{rank}, L_{gan}, (3) and (4) refer to Eq. (6.6), Eq. (6.7), Eq. (6.4), and Eq. (6.5), respectively.

prediction on the generated pairs. (3) RelGAN manifests a three-step like curve, which indicates a failure of fine-grained generation. This is mainly because of its specific design of the interpolation loss. (4) FN presents a linear tendency like TRIP, which denotes that it can make a fine-grained control over the attribute. However, the mean of the ranking output for the generated pairs is relatively low in FN, since it fails to translate some input images into the desired output images. This is verified by its low translation accuracy (Fig. 6.9), lower than 85%. In addition, FN also exhibits a large variance of the ranking output due to the poor image quality.

6.4.4 Ablation Study

In Fig. 6.12 and Table 6.3, we show an ablation study of our model. (1) Without (w/o) L_{rank}, the generated images exhibit no change over the input image. The generator fails to learn the translation function, which is demonstrated by an extremely low translation accuracy (AAS). (2) TRIP with only the vanilla ranking loss, i.e., w/o L_{gan} & $\lambda = 0$, performs poor in terms of both the realistic quality and the attribute-specified quality. It produces unrealistic images, with a very high FID. Meanwhile, the generated images change in terms of another aspect (the background color of images) rather than the "smile" attribute along with the controlled variable, with a low AAS. (3) TRIP only with the adversarial ranking loss, i.e., w/o L_{gan}, yields relatively realistic images. The generated images change exclusively in terms of the "smile" attribute along with the controlled variable. However, this model is still inferior over the model with L_{gan} (i.e., TRIP) w.r.t. the realistic quality. (4) Setting $\lambda = 0$, i.e., without considering the adversarial ranking, the performance of facial image manipulation collapses, obtaining a low translation accuracy (AAS). (5) When optimizing with Eq. 6.4, i.e., not linearing the ranking output for the generated pairs, the fine-grained control over the attributes fails, getting a high DSSIM score. (6) With our TRIP, the generated images present desired changes consistent with the latent variable and possess high quality.

Table 6.3 Quantitative evaluation of ablation study on CelebA-HQ ("smile" attribute). The models in columns 2 to 7 correspond to the models in rows 1 to 6 of Fig. 6.12, respectively. without (w/o); with (w).

Model	w/o L_{rank}	w/o L_{gan}&$\lambda = 0$	w/o L_{gan}	$\lambda = 0$	CLS (3)	TRIP
AAS	9.73	46.2	98.47	55.7	92.37	97.69
DSSIM	1.51E-05	0.0016	0.0058	0.0011	0.0163	0.0030
FID	29.00	428.35	78.08	8.55	11.33	10.19

6.4.5 Extension to Multiple Attributes

We conduct fine-grained I2I translation with both "smile" and "male" on CelebA-HQ to show that TRIP can generalize well to multiple attributes. A two-dimension variable is used to control the change of both attributes, simultaneously.

The generated images conditioning on different $\vec{v}$ are shown in Fig. 6.13. (1) TRIP can disentangle multiple attributes. When $\vec{v} = [-1,0]/[1,0]$, the generated images $O_{-1,0}/O_{1,0}$ appear "less smiling"/"more smiling" with no change on the "male" attribute. When $\vec{v} = [0,-1]/[0,1]$, the generated images $O_{0,-1}/O_{0,1}$ appear "less masculine"/"more masculine" with no change in the "smile" attribute. In addition, a fine-grained control over the strength of a single attribute is still practical. (2) TRIP can manipulates the subtle changes of multiple

attributes simultaneously. For example, when conditioning $\vec{v} = [1, -1]$, the generated image $O_{1,-1}$ appear "less smiling" and "more masculine".

6.4.6 Convergence of TRIP

TRIP converges to an equilibrium when the generator produces the realistic image with desired changes over the input image regarding the target attribute, and the ranker makes reliable prediction for the difference between the translated images and the input image w.r.t the target attribute.

To justify our claim, we plot the training curve of the ranker and the generator, respectively, as shown in Fig. 6.14. It demonstrates that the ranker and the generator are trained against each other until convergence.

Further, we plot the distribution of the ranker's prediction for real image pairs and generated image pairs with different RAs $r \in \{+1/0/-1\}$ using the ranker in Fig. 6.15. (1) At the beginning of the training, the ranker gives similar predictions for real image pairs with different RAs. The same observations can also be found on the generated image pairs. (2) After 100 iterations, the ranker learns to give the desired prediction for different kinds of pair, i.e., > 0 (averaged) for pairs with RA $(+1)$, 0 (averaged) for pairs with RA (0) and < 0

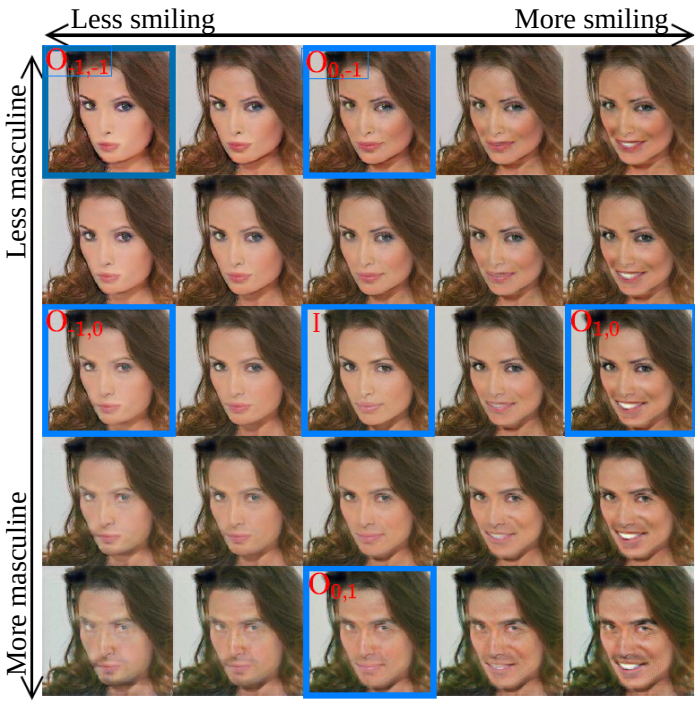


Figure 6.13 Fine-grained I2I translation with "smile" and "male" attributes. The middle is the input image.

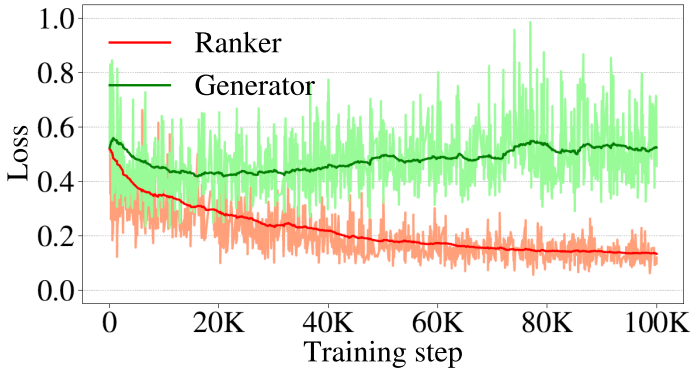


Figure 6.14 The curve of training loss. The ranker and the generator are trained against each other until convergence. Ranker loss $= L^R_{rank} + \lambda_g L^R_{gan} + \lambda_{gp} L_{gp}$. Generator loss $= L^G_{rank} + \lambda_g L^G_{gan} + \lambda_{gp} L_{cycle}$.

(averaged) for pairs with RA (-1). (3) After $9,900$ iterations, TRIP converges. In terms of the real image pairs, the ranker output $+1$ for the pairs with RA $(+1)$, 0 for the pairs with RA (0) and -1 for the pairs with RA (-1) in the sense of average. This verifies that our ranker can give precise ranking predictions for real image pairs. In terms of the generated pairs, the ranker outputs $+0.5$ for the pairs with RA $(+1)$, 0 for the pairs with RA (0) and -0.5 for the pairs with RA (-1) in the sense of average. This is a convergence state due to rival preferences. We take pairs with RA $(+1)$ as an example. The generated pairs with RA $(+1)$ are expected to be assigned 0 when optimizing the ranker, and to be assigned $+1$ when optimizing the generator. Therefore, the convergence state should be around 0.5.

6.5 Summary

This chapter proposes TRIP for high-quality fine-grained I2I translation. TRIP elegantly reconciles the goal for fine-grained translation and the goal for high-quality generation through adversarial ranking. It broadens the principle of adversarial training by extending the adversarial game on the classification to ranking. The adversarial ranking between the ranker and the generator is defined on comparisons between pairs of samples, which can trigger a high-quality generation conditioned on a continuous variable. By contrast, the adversarial classification between the discriminator and the generator is defined on single samples (or together with labels), which can trigger a high-quality unconditional generation (or a high-quality generation conditioned on a discrete variable).

Since the supervised pairwise ranking and the unsupervised generation target are incorporated into a single model function, TRIP can be deemed as a new form of semi-supervised

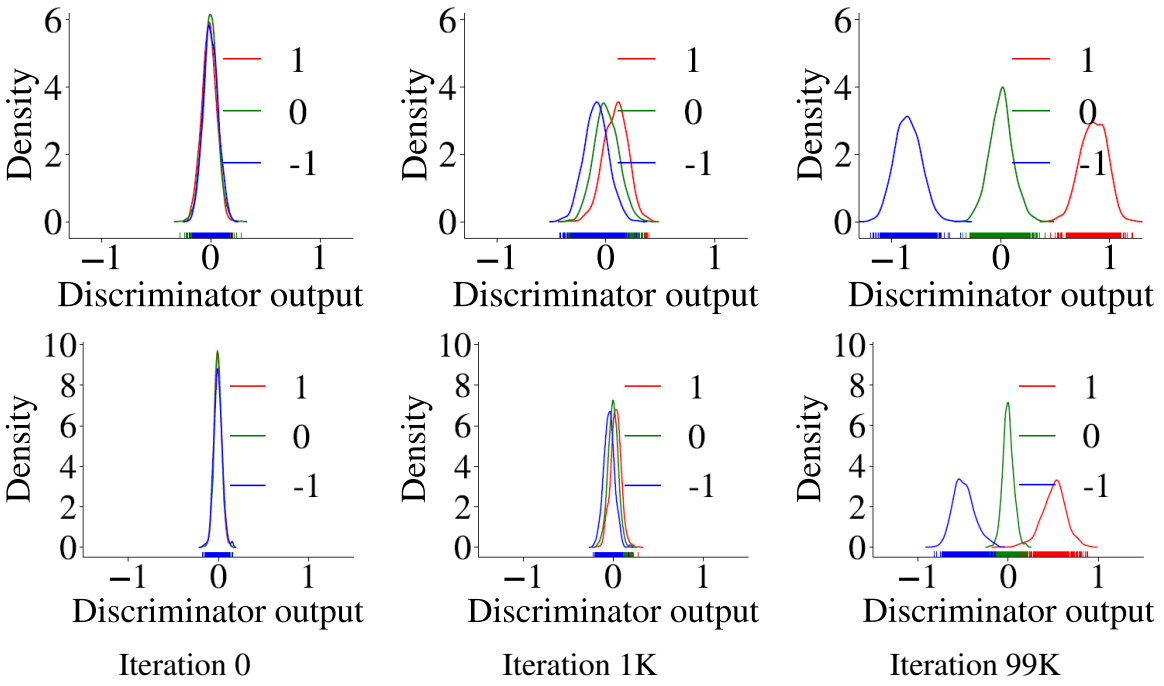


Figure 6.15 The density plot of the ranker's output for real image pairs (first row) and generated pairs (second row) with different relative attributes $r \in \{+1/0/-1\}$, respectively, during the training process.

GAN [Odena, 2016]. The empirical experiments demonstrate that TRIP achieves state-of-art results in generating high-fidelity images that exhibit smooth changes exclusively w.r.t. the specified attributes according to the fine-grained score and the quality score. One interesting future direction is to extend our idea to semi-supervised ranking [Duh and Kirchhoff, 2011] given that it has been theoretically proven that semi-supervised GAN can improve the performance of semi-supervised classification [Dai et al., 2017].

Chapter 7

Conclusion

Generative modeling from human preferences that aims to generate data meeting user needs can be applied to various application fields, such as inverse generative design [Sanchez-Lengeling and Aspuru-Guzik, 2018], creative art [Bernardo et al., 2016]. Inspired by real-world problems, this thesis mainly studies desired data generation at the dataset level and at the instance level, respectively. In addition, we conduct two preliminary studies on DGMs, which solve the mode collapse issue in GANs and propose a disentangled VAE framework to improve clustering. The overall work of this thesis is present in Fig. 1.2. In the following, we summarize the main contributions of this thesis.

The first work to apply human preferences for dataset-level desired data generation. Unlike existing work that guides the DGMs by labels or evaluators, which requires complete knowledge about the properties that is expensive to access in real-world applications, preferences are simply partial knowledge about the properties and can be more readily available. Our proposed work DiCGAN in Chapter 4 incorporates preferences directly into the data generation based on the insight of considering the critic values as the ranking scores that represent user preferences, which results in biasing the generative model towards the desired data distribution. The implementation of DiCGAN is simply incorporating an additional ranking loss based on preferences into the critic of WGAN, which can also be extended to other critic-based GAN frameworks. Furthermore, the reformulation of DiCGAN as a constrained optimization problem inspires a training paradigm with multi-step distribution shift and the theoretical proof of the convergence to the desired data distribution. Empirical studies demonstrate that our DiCGAN can generate images to meet the user's interest on MNIST and CelebAHQ and help design biological products with desired properties on the gene sequence dataset, outperforming than various baselines that relies on labels/evaluator especially in the cases of insufficient desired data and limited supervision.

A novel generative adversarial learning paradigm that can learn the distribution of a user-specified subset of training data in a single step with theoretical guarantees. For a more efficient way of generative modeling from human preferences in the dataset-level desired data generation problem, we further propose a generative adversarial ranking framework GARNet in Chapter 5, which consists of an adversarial ranking between a ranker and a generator. Consequently, GARNet optimizes a relativisitic f-divergence between the desired data distribution and the generated data distribution. This reveals a new generative modeling under the context of human preferences, which can directly enable a distribution alignment between the generated data and the desired data (part of the training data). We empirically justify our GARNet can learn a desired data distribution (determined by the given score vector) from full preferences in terms of discrete and continuous properties. We further show that our GARNet can better retrieve the desired data distribution from partial preferences than existing (conditional) GAN variants on three image datasets. We also validate the advantage of preference-guided GARNet over GAN conditioned on labels in imbalanced learning.

The first framework that reconciles the modeling for preferences and data quality for desired data generation at the instance level. In particular, we consider the fine-grained image-to-image translation task. Existing work suffers from the conflict between the goal for fine-grained translation and the goal for high-quality generation. We proposed TRIP to coordinate these two goals in Chapter 6. TRIP inherits the idea of generative adversarial ranking paradigm from GARNet, playing an adversarial game between a ranker and a generator. Such a adversarial ranking game is beneficial for triggering a high-quality generation conditioned on a continuous variable in the scenario of instance-level desired data generation, while the adversarial classification game usually triggers a high-quality generation conditioned on a discrete variable. In addition, our tailor-designed objective of the generator, which enforces ranking predictions consistent with the conditioned continuous attribute variable, promotes a better fine-grained control over the interested attribute. Experiments on two face image datasets and one shoe dataset in terms of single/multiple attributes show that our TRIP outperforms existing baselines in both generation quality and fine-grained translation performance.

Finally, we discuss some future directions. (1) Instance-based human preferences. In some real-world applications, it is nontrivial to describe desired designs in terms of specified properties. Instead, a set of instances are often provided as exemplars to conceptualize the desired properties. Based on information retrieval [Liu et al., 2009], the given instances can be regarded as queries to construct ranking lists. This would be a new kind of instance-level desired data generation that takes an instance as a condition to generate data that possess

the similar properties as the conditioned instance. (2) Generative modeling from a mixture of preferences. Our frameworks are basically modeled under the homogeneity assumption of rank aggregation that only one ground truth full ranking exists [Lin, 2010]. When preferences are sampled from heterogeneous user groups, it is supposed to consider mixture rank aggregation [Zhao et al., 2016]. We conduct a preliminary exploration for a simple case where the group affiliation of each preference is known (Section 5.6.3 and Section 6.4.5). We leave the general case of a mixture of preferences for future work. (3) Preference-guided diffusion model. As diffusion models have recently been growing popular generative models due to their stable training and high-quality generation capabilities [Croitoru et al., 2023], it is promising to consider human preferences under this framework. One simple idea is to apply a ranker to guide the generation to satisfy the desired properties. But the trade-off between sample quality and diversity may arise like classifier-guided diffusion model [Ho and Salimans, 2022], which needs more explorations.

Appendix A

Appendix for Chapter 3

A.1 Proof of Theorem 1 in COUF

Proof. Based on the definition of mutual information, we have

$$\begin{aligned} I(z,\omega) &= \int P(z,\omega)\log\frac{P(z,\omega)}{P(z)P(\omega)}dzd\omega \\ &= \int P(z,\omega)\log\frac{P(\omega\mid z)}{P(\omega)}dzd\omega. \end{aligned}$$

Assume $P(x,c,z,\omega) = P(x,c)P(z\mid x,c)P(\omega\mid x,c,z) = P(x,c)P(z\mid x,c)P(\omega\mid x,c)$, where $P(\omega\mid x,c,z) = P(\omega\mid x,c)$ follows the conditional independence. Since $P(\omega\mid z) = \int P(x,c,\omega\mid z)dxdc = \int \frac{P(z|x,c)P(x,c)}{P(z)}P(\omega\mid x,c)dxdc$ is intractable, we introduce an auxiliary distribution $Q(\omega\mid z)$ as an approximation to $P(\omega\mid z)$ [Alemi et al., 2017]. Because

$$\mathrm{KL}[P(\omega\mid z)||Q(\omega\mid z)] \geq 0 \Longrightarrow \int P(\omega\mid z)\log P(\omega\mid z)d\omega \geq \int P(\omega\mid z)\log Q(\omega\mid z)d\omega,$$

we obtain

$$\begin{aligned} I(z,\omega) &\geq \int P(z,\omega)\log\frac{Q(\omega\mid z)}{P(\omega)}dzd\omega \\ &= \int P(z,\omega)\log Q(\omega\mid z)dzd\omega - \int P(\omega)\log P(\omega)d\omega \\ &= \int P(z,\omega)\log Q(\omega\mid z)dzd\omega + H(\omega). \end{aligned}$$

$H(\omega)$ can be ignored as it is independent of our optimization procedure. We can rewrite $P(z,\omega) = \int P(x,c,z,\omega)dxdc = \int P(x,c)P(z \mid x,c)P(\omega \mid x,c)dxdc$. Then we have

$$\begin{aligned} I(z,\omega) &\geq \int P(x,c)P(z \mid x,c)P(\omega \mid x,c)\log Q(\omega \mid z)dxdcdzds \\ &= \mathbb{E}_{(x,c)\sim P(x,c)}\mathbb{E}_{z\sim P(z|x,c)}\mathbb{E}_{s\sim P(\omega|x,c)}\log Q(\omega \mid z)d\omega \\ &= L_I(z,\omega). \end{aligned}$$

The auxiliary distribution $Q(\omega \mid z)$ can be naturally defined by our k-means clustering module (Section 3.2.1). Accordingly, we have

$$Q(\omega_{nk} = 1 \mid z_n) = \frac{\exp\left(-\tau \left\| z_n - e_k \right\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(-\tau \left\| z_n - e_i \right\|_2^2\right)}. \tag{A.1}$$

Note that we approximate the posterior $P(z \mid x,c)$ by the VAE encoder $Q(z \mid x)$ constrained with the minimization of $I(z,c)$ and usually one particle z_n is sampled from $Q(z|x)$ to reconstruct x_n [Kingma and Welling, 2014]. Together with the given clustering assignment $\omega_n \sim P(\omega \mid x,c)$, we have

$$L_I(z,\omega) = \sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk} \log \frac{\exp\left(-\tau \left\| z_n - e_k \right\|_2^2\right)}{\sum_{i=1}^{K} \exp\left(-\tau \left\| z_n - e_i \right\|_2^2\right)}, \tag{A.2}$$

where $k = 1,2,\ldots K$. The value of Eq. (A.1) approaches zero for all k except for the one corresponding to the smallest distance [Bishop, 2006, Kulis and Jordan, 2012]. Thus we have

$$Eq.\ (\text{A.2}) \underset{\tau \longrightarrow +\infty}{\Longrightarrow} -\sum_{n=1}^{N}\sum_{k=1}^{K} \omega_{nk} \left\| z_n - e_k \right\|_2^2.$$

Therefore, we can obtain

$$\mathcal{L}_{\text{cluster}} \geq -I(z,\omega), \tag{A.3}$$

which completes the proof. □

A.2 More Experimental Setup of COUF

We introduce five image datasets in details as follows:

1. The face dataset from the UCI KDD repository, called *UCI-Face* [Bay et al., 2000], contains face images of 20 people in four different poses, in which the face identity is

set as the unwanted factor. Clustering on face poses while ignoring face identity can be applied in applications such as pose recognition [Liu et al., 2010].

2. The Rotated Fashion-MNIST dataset, called *Rotated Fashion*, is constructed by introducing the rotation factor into the Fashion-MNIST dataset [Xiao et al., 2017], which has 10 classes of objects. Particularly, we pick up images from the cloth categories, i.e., "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat" and "Shirt", for simplicity. We first randomly sample $1,000$ images from each of the six classes (zero degree). Then, each image is augmented with four views of 72, 144, 216, and 288 degrees, respectively. Thus, there are five categories of the rotation factor and $30,000$ images in total. The cloth category is regarded as the unwanted factor. Clustering in terms of cloth rotation can be applied in applications that emphasize cloth diversity [Sun et al., 2020, Ma et al., 2020].

3. *MNIST-USPS* is the union of all training samples from MNIST [Lecun et al., 1998] and USPS [Hull, 1994], consisting of digit zero to nine. The source of digits is the unwanted factor.

4. *Office-31* [Saenko et al., 2010] consists of images with 31 different categories collected from three distinct domains: Amazon, Webcam, and DSLR. Each domain contains all the categories but with different shooting angles, lighting conditions, or forms of presentation, etc. We select samples from Amazon and Webcam as training data following [Li et al., 2020a]. The source of collected data is regarded as the unwanted factor.

5. The corrupted CIFAR10 dataset, named *CIFAR10-C* [Hendrycks and Dietterich, 2019], contains image corruptions for CIFAR10 images with 10 object classes. There are four main categories of corruption and 15 fine-grained categories: Weather (frost, fog, snow, spatter), Blur (Gaussian, zoom, defocus, glass, motion), Noise (impulse, shot, Gaussian, speckle), and Digital (elastic transform, JPEG, pixelation, brightness, contrast, saturate). Particularly, we consider one in each main category of corruptions, namely, frost, Gaussian blur, impulse noise, and elastic transform for simplicity. The corruption type is regarded as the unwanted factor.

Appendix B

Appendix for Chapter 4

B.1 Proof of Proposition 3 in DiCGAN

Proof. Without loss of generality, we represent $P_{\mathrm{r}}(x)$ and $P_{\mathrm{g}}(x)$ in a fine-grain formulation. Namely,

$$P_{\mathrm{r}}(x) = (1-\alpha)P_{\mathrm{d}}(x) + \alpha P_{\mathrm{u}}(x), \quad P_{\mathrm{g}}(x) = P_{\mathrm{d}}(x), \tag{B.1}$$

where $\alpha \in [0,1]$ is a very small value such as $d\left(P_{\mathrm{r}}(x), P_{\mathrm{d}}(x)\right) \leq \varepsilon$ is satisfied.

Furthermore, according to our definition of the ranking model, the score for the desired data should be higher than that of the undesired data, namely

$$D(x) \begin{cases} > T, & \text{if } x \sim P_{\mathrm{d}}(x); \\ \leq T, & \text{if } x \sim P_{\mathrm{u}}(x), \end{cases} \tag{B.2}$$

where $o_{min} < T < o_{max}$. (1) (o_{min}, o_{max}) is introduced since the critic score is always bounded; (2) T denotes some value to discriminate the desired data from undesired data.

Taking into consideration of both Eq. (B.1), (B.2), we have

$$\begin{aligned} \mathbb{E}_{P_{\mathrm{r}}(x)}\left[D\left(x\right)\right] &= \int \left[(1-\alpha)P_{\mathrm{d}}(x) + \alpha P_{\mathrm{u}}(x)\right] D(x)dx \\ &= \int \left[(1-\alpha)P_{\mathrm{d}}(x)\right] D(x)dx + \int \left[\alpha P_{\mathrm{u}}(x)\right] D(x)dx \\ &= \int P_{\mathrm{d}}(x)D(x)dx - \alpha\left(\int P_{\mathrm{d}}(x)D(x)dx - \int P_{\mathrm{u}}(x)D(x)dx\right). \end{aligned} \tag{B.3}$$

Considering that (1) $P_{\mathrm{d}}(x)$ and $P_{\mathrm{u}}(x)$ are always positive; (2) $D(x)$ is continuous and bounded on the domain of x with respect to $P_{\mathrm{d}}(x)$ and $P_{\mathrm{u}}(x)$, respectively, we have the following

derivations according to the mean value theorem for integrals:

$$\exists\, \xi_{\mathrm{d}} \in (T, o_{max}], \quad \int P_{\mathrm{d}}(x)D(x)dx = \xi_{\mathrm{d}} \int P_{\mathrm{d}}(x)dx = \xi_{\mathrm{d}};$$
$$\exists\, \xi_{\mathrm{u}} \in [o_{min}, T], \quad \int P_{\mathrm{u}}(x)D(x)dx = \xi_{\mathrm{u}} \int P_{\mathrm{u}}(x)dx = \xi_{\mathrm{u}}.$$

Since $\xi_{\mathrm{d}} > \xi_{\mathrm{u}}$, we have

$$\alpha \left(\int P_{\mathrm{d}}(x)D(x)dx - \int P_{\mathrm{u}}(x)D(x)dx \right) = \alpha(\xi_{\mathrm{d}} - \xi_{\mathrm{u}}) > 0.$$

$\Longrightarrow$ Eq. (B.3) $= \mathbb{E}_{P_{\mathrm{d}}(x)}[D(x)] - \delta$, where $\delta = \alpha(\xi_{\mathrm{d}} - \xi_{\mathrm{u}}) > 0$. Furthermore, by replacing $P_{\mathrm{d}}(x)$ with $P_{\mathrm{g}}(x)$, we have

$$\mathbb{E}_{P_{\mathrm{g}}(x)}[D(x)] = \mathbb{E}_{P_{\mathrm{r}}(x)}[D(x)] + \delta,$$

for some $\delta > 0$. □

B.2 Proof of Proposition 4 in DiCGAN

Proof. The major correction is divided into a sequence of minor corrections. In the first minor correction, the training data distribution is $P_{\mathrm{r}}(x)$. Derived by Corollary 2, after this minor correction, we have

$$d(P_{\mathrm{r}}, P_{\mathrm{d}}) - d(P_{\mathrm{g}}^1, P_{\mathrm{d}}) = \delta.$$

With $d(P_{\mathrm{r}}, P_{\mathrm{d}}) = T_0$, we can get

$$d(P_{\mathrm{g}}^1, P_{\mathrm{d}}) = T_0 - \delta.$$

In the second minor correction, we can replace all training samples with the generated samples obtained in the first correction. Thus, the training data distribution becomes $P_{\mathrm{g}}^1(x)$. After two minor corrections, similarly, we can get

$$d(P_{\mathrm{g}}^2, P_{\mathrm{d}}) = T_0 - 2\delta.$$

So on and so forth. After k minor corrections, we can get

$$d(P_{\mathrm{g}}^k, P_{\mathrm{d}}) = T_0 - k\delta.$$

□

Appendix C

Appendix for Chapter 5

C.1 Proof of Theorem 3 in GARNet

Proof of Eq. (5.17) being a relativistic f-divergence.

Proof. Recalling Eq. (C.9), we have

$$
\begin{aligned}
D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) &= \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S}\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\mathcal{L}_{CE}\left(\pi\big(s^{(R)}\big), R\big(s^{(R)}\big)\right)\right] \\
&\longrightarrow \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\log \frac{1}{c + e^{-\left(\left(\sum_{i=1}^T q_i R(s^i)\right) - R\left(x_{\mathrm{g}}\right)\right)}}\right] \\
&\overset{①}{=} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\log \frac{1}{c + e^{-\left(\left(\sum_{i=1}^T q_i R(s^i)\right) - R\left(x_{\mathrm{g}}\right)\right)}} + \log(c+1)\right] \\
&\overset{②}{=} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f\left(\sum_{i=1}^T q_i R(s^i) - R(x_{\mathrm{g}})\right)\right],
\end{aligned}
\tag{C.1}
$$

where ① is valid due to the addition of a constant. ② is obtained by denoting $f = \log\frac{1}{c+e^{-z}} + \log(c+1)$. Accordingly, f is a concave function; $f(0) = 0$; f is differentiable at 0; $f'(0) \neq 0$; $\sup_x f(x) = M > 0$; and $\arg\sup_x f(x) > 0$.

Let $R^{\mathrm{w}}(x) = C' \ \forall x \in \mathcal{X}$ (worst possible choice of R), where C' is a constant.
Let $R^* = \arg\sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f\left(\sum_{i=1}^T q_i R(s^i) - R(x_{\mathrm{g}})\right)\right]$ be the best possible choice of R.

#1 Prove that $D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) \geq 0$.

$$D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) = \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f(\sum_{i=1}^T q_i R^*(s^i) - R^*(x_{\mathrm{g}}))\right] \geq \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f(\sum_{i=1}^T q_i R^{\mathrm{w}}(s^i) - R^{\mathrm{w}}(x_{\mathrm{g}}))\right] \tag{C.2}$$
$$\xrightarrow[①]{\sum_{i=1}^T q_i \to 1} D_f(P_u, P_g) \geq 0.$$

① is based on Eq. (C.6b).

#2 Prove that $P_{\mathrm{d}} = \sum_{i=1}^T q_i P_i = P_{\mathrm{g}} \Longrightarrow D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) = 0$.

$$\begin{aligned}
D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) &= \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f\left(\sum_{i=1}^T q_i R^*(s^i) - R^*(x_{\mathrm{g}})\right)\right] \overset{①}{\leq} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T}} \left[f\left(\sum_{i=1}^T q_i R^*(s^i) - \mathop{\mathbb{E}}_{x_{\mathrm{g}}\sim P_{\mathrm{g}}}\left[R^*(x_{\mathrm{g}})\right]\right)\right] \\
&\overset{②}{\leq} f\left(\mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T}} \left[\sum_{i=1}^T q_i R^*(s^i)\right] - \mathop{\mathbb{E}}_{x_{\mathrm{g}}\sim P_{\mathrm{g}}}\left[R^*(x_{\mathrm{g}})\right]\right) = f\left(\sum_{i=1}^T q_i \mathop{\mathbb{E}}_{s^i\sim P_i}\left[R^*(s^i)\right] - \mathop{\mathbb{E}}_{x_{\mathrm{g}}\sim P_{\mathrm{g}}}\left[R^*(x_{\mathrm{g}})\right]\right) \\
&\overset{③}{=} f\left(\sum_{i=1}^T q_i \mathop{\mathbb{E}}_{x\sim P_i}[R^*(x)] - \mathop{\mathbb{E}}_{x\sim P_{\mathrm{g}}}[R^*(x)]\right) = f\left(\int R^*(x)\sum_{i=1}^T q_i P_i dx - \int R^*(x) P_{\mathrm{g}} dx\right) \\
&\overset{④}{=} f(0) = 0,
\end{aligned} \tag{C.3}$$

where ① and ② follow Jensen's inequality for concave function f. ③ is valid due to change of variables. ④ is valid because $P_{\mathrm{g}} = \sum_{i=1}^T q_i P_i$. Since $D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) \geq 0$ (Eq. (C.2)), we have $D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) = 0$.

#3 Prove that $D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) = 0 \Longrightarrow P_{\mathrm{d}} = \sum_{i=1}^T q_i P_i = P_{\mathrm{g}}$.

We prove this by contraposition. Namely, we prove that $P_{\mathrm{d}} \neq P_{\mathrm{g}} \Longrightarrow D_f(P_{\mathrm{d}}, P_{\mathrm{g}}) \neq 0$.
Let $\mathcal{H} = \{x | P_{\mathrm{d}}(x) > P_{\mathrm{g}}(x)\}$. Since $P_{\mathrm{d}} \neq P_{\mathrm{g}}$, we have $\mathcal{H} \neq \emptyset$.
Let $u = \int_{\mathcal{H}} P_{\mathrm{d}}(x)dx \Longrightarrow (1-u) = \int_{\mathcal{X}\setminus\mathcal{H}} P_{\mathrm{d}}(x)dx$. Let $v = \int_{\mathcal{H}} P_{\mathrm{g}}(x)dx \Longrightarrow (1-v) = \int_{\mathcal{X}\setminus\mathcal{H}} P_{\mathrm{g}}(x)dx$.
Then, we have $u > 0, v > 0$, and $u > v$.

Let $R'(x) = \begin{cases} \Delta & \text{if } x \in \mathcal{H} \\ 0 & \text{else} \end{cases}$, where $\Delta \neq 0$.

Let $L(\Delta) = \underset{\substack{s^1 \sim P_1 \\ \cdots \\ s^T \sim P_T \\ x_g \sim P_g}}{\mathbb{E}} \left[f\left(\sum_{i=1}^{T} q_i \left(R'(s^i) - R'(x_g) \right) \right) \right]$. Then,

$$
\begin{aligned}
L(\Delta) &= \underset{\substack{s^1 \sim P_1 \\ \cdots \\ s^T \sim P_T \\ x_g \sim P_g}}{\mathbb{E}} \left[f\left(\sum_{i=1}^{T} q_i \left(R'(s^i) - R'(x_g) \right) \right) \right] \overset{①}{\geq} \underset{\substack{s^1 \sim P_1 \\ \cdots \\ s^T \sim P_T \\ x_g \sim P_g}}{\mathbb{E}} \left[\sum_{i=1}^{T} q_i f(R'(s^i) - R'(x_g)) \right] \\
&= \sum_{i=1}^{T} \underset{\substack{s^i \sim P_i \\ x_g \sim P_g}}{\mathbb{E}} \left[q_i f(R'(s^i) - R'(x_g)) \right] \overset{②}{=} \sum_{i=1}^{T} \underset{\substack{x_d \sim P_i \\ x_g \sim P_g}}{\mathbb{E}} \left[q_i f(R'(x_d) - R'(x_g)) \right] \\
&= \int_{\mathcal{X}} \int_{\mathcal{X}} f(R'(x_d) - R'(x_g)) (\sum_{i=1}^{T} q_i P_i(x_d)) P_g(x_g) dx_d dx_g \overset{③}{=} \underset{\substack{x_d \sim P_d \\ x_g \sim P_g}}{\mathbb{E}} \left[f(R'(x_d) - R'(x_g)) \right] \\
&= \int_{\mathcal{H}} \int_{\mathcal{H}} f(R'(x_d) - R'(x_g)) P_d(x_d) P_g(x_g) dx_d dx_g + \int_{\mathcal{H}} \int_{\mathcal{X} \setminus \mathcal{H}} f(R'(x_d) - R'(x_g)) P_d(x_d) P_g(x_g) dx_d dx_g \\
&+ \int_{\mathcal{X} \setminus \mathcal{H}} \int_{\mathcal{H}} f(R'(x_d) - R'(x_g)) P_d(x_d) P_g(x_g) dx_d dx_g \\
&+ \int_{\mathcal{X} \setminus \mathcal{H}} \int_{\mathcal{X} \setminus \mathcal{H}} f(R'(x_d) - R'(x_g)) P_d(x_d) P_g(x_g) dx_d dx_g,
\end{aligned}
\tag{C.4}
$$

① follows Jensen's inequality for concave function f. ② is valid due to the change of variables. ③ is obtained by denoting $P_d = \sum_{i=1}^{t} q_i P_i$.

Since $u(1-v) > v(1-u)$, we have that $\exists \Delta^* > 0 s.t. L(\Delta^*) > 0$ according to Lemma A.3 in Jolicoeur-Martineau [2020]. Thus, if we let $\Delta = \Delta^*$, we have

$$
\begin{aligned}
&\text{Eq. (C.4)} > 0 \xrightarrow[①]{\sum_{i=1}^{T} q_i \to 1} \\
&D_f(P_u, P_g) = \underset{\substack{s^1 \sim P_1 \\ \cdots \\ s^T \sim P_T \\ x_g \sim P_g}}{\mathbb{E}} \left[f(\sum_{i=1}^{T} q_i R^*(s^i) - R^*(x_g)) \right] \geq \underset{\substack{s^1 \sim P_1 \\ \cdots \\ s^T \sim P_T \\ x_g \sim P_g}}{\mathbb{E}} \left[f(\sum_{i=1}^{T} q_i R'(s^i) - R'(x_g)) \right] > 0
\end{aligned}
\tag{C.5}
$$

① is based on Eq. (C.6b). □

C.2 Proof of Proposition 5 in GARNet

Proof. First of all, given the definition of $\pi\left(s^{(R)}\right)$, we have

$$\lim_{\frac{1}{e^{a-b}}\to 0} q_{T+1} = \lim_{\frac{1}{e^{a-b}}\to 0} \sigma\left(\pi\left(s^{(R)}\right)\right)_{T+1} = \lim_{\frac{1}{e^{a-b}}\to 0} \frac{e^b}{e^b + \sum_{j=1}^{T} e^{a+(T-j)d}} \overset{①}{=} 0, \tag{C.6a}$$

$$\lim_{\frac{1}{e^{a-b}}\to 0} \sum_{i=1}^{T} q_i = \lim_{\frac{1}{e^{a-b}}\to 0} 1 - q_{T+1} \overset{②}{=} 1. \tag{C.6b}$$

where ①, ② follow the Squeeze theorem [Stewart et al., 2020] given $0 < q_{T+1} < \frac{1}{e^{a-b}}$ and $\frac{1}{e^{a-b}} \to 0$.

According to Eq. (5.5), we have

$$\begin{aligned}
\mathcal{L}_{L2R}\left(\pi\left(s^{(R)}\right), R\left(s^{(R)}\right)\right) &= \sum_{i=1}^{T+1} \sigma(\pi(s^{(R)}))_i \log \sigma(R(s^{(R)}))_i \\
&\overset{①}{=} \left(\sum_{i=1}^{T} q_i \log \frac{e^{R(s^i)}}{e^{R(x_g)} + \sum_{j=1}^{T} e^{R(s^j)}}\right) + q_{T+1} \log \frac{e^{R(x_g)}}{e^{R(x_g)} + \sum_{j=1}^{T} e^{R(s^j)}} \\
&= \log \frac{e^{\sum_{i=1}^{T} q_i R(s^i)}}{\left(e^{R(x_g)} + \sum_{j=1}^{T} e^{R(s^j)}\right)^{1-q_{T+1}}} + q_{T+1} \log \frac{e^{R(x_g)}}{e^{R(x_g)} + \sum_{j=1}^{T} e^{R(s^j)}} \\
&\overset{②}{=} \log \frac{1}{e^{R(x_g) - \sum_{i=1}^{T} q_i R(s^i)} + \sum_{j=1}^{T} e^{R(s^j) - \sum_{i=1}^{T} q_i R(s^i)}}.
\end{aligned} \tag{C.7}$$

① follows the definition of the target preference $s^{(R)}$ in Eq. (5.6a). ② is valid due to Eq. (C.6a) and Eq. (C.6b).

Meanwhile, when the ranker is approaching the optima R^* for a fixed generator G, i.e., $\sigma\left(R^*\left(s^{(R)}\right)\right) = \sigma\left(\pi\left(s^{(R)}\right)\right)$, we have $R^*(s^{(R)}) = \pi\left(s^{(R)}\right) + \delta$ due to the translation invariance of softmax [Laha et al., 2018], i.e., $\sigma(r+\delta) = \sigma(r)$. Therefore, $R^*(s^{(R)})$ is also arithmetic progression with a common difference of $-d$ same as $\pi\left(s^{(R)}\right)$. Then, we have

$$\begin{aligned}
R^*(s^j) - \sum_{i=1}^{T} q_i R^*(s^i) &= \left[R^*(s^1) - (j-1)d\right] - \sum_{i=1}^{T} q_i \left[R^*(s^1) - (i-1)d\right] \\
&= R^*(s^1) - \left(\sum_{i=1}^{T} q_i\right) R^*(s^1) - \left(j - 1 + \sum_{i=1}^{T} q_i(1-i)\right) d \\
&\overset{①}{=} \left(\sum_{i=1}^{T} q_i i - j\right) d = c_j.
\end{aligned} \tag{C.8}$$

where c_j is a constant $\forall j = 1,2,\ldots,T$, exclusively determined by the pre-specified score vector $\pi(s)$. ① is valid due to Eq. (C.6b).

Therefore, the objective for the ranker R can be approximated as follows:

$$
\begin{aligned}
Eq.\ (5.7\text{a}) &\overset{①}{\approx} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{S}\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\log \frac{1}{c+e^{-(\sum_{i=1}^{T} q_i R(s^i)-R(x_{\mathrm{g}}))}}\right] \\
&\overset{②}{=} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\log \frac{1}{c+e^{-\left(\left(\sum_{i=1}^{T} q_i R(s^i)\right)-R\left(x_{\mathrm{g}}\right)\right)}}\right] \\
&\overset{③}{=} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[\log \frac{1}{c+e^{-\left(\left(\sum_{i=1}^{T} q_i R(s^i)\right)-R\left(x_{\mathrm{g}}\right)\right)}}+\log(c+1)\right] \\
&\overset{④}{=} \sup_{R:\mathcal{X}\to\mathbb{R}} \mathop{\mathbb{E}}_{\substack{s^1\sim P_1\\ \cdots\\ s^T\sim P_T\\ x_{\mathrm{g}}\sim P_{\mathrm{g}}}} \left[f\left(\sum_{i=1}^{T} q_i R(s^i)-R(x_{\mathrm{g}})\right)\right],
\end{aligned}
\tag{C.9}
$$

① is reasonable because $\sum_{j=1}^{T} e^{R(s^j)-\sum_{i=1}^{T} q_i R(s^i)}$ in Eq. (C.7) would degenerate to a constant $c = e^{c_1+c_2+\ldots+c_T}$ following Eq. (C.8), being independent of the ranker when the ranker is approaching the optima. ② is valid as $y(s^i) \equiv y_i$ according to Eq. (5.4). ③ is valid due to the addition of a constant. ④ is obtained by denoting $f(z) = \log \frac{1}{c+e^{-z}} + \log(c+1)$.

Note f is a concave function; $f(0) = 0$; f is differentiable at 0; $f'(0) \neq 0$; $\sup_x f(x) = M > 0$; and $\arg\sup_x f(x) > 0$. According to Theorem 3, the optimal ranker of our GARNet approximately estimates a relativistic f divergence. □

Appendix D

Appendix for Chapter 6

D.1 More Experimental Setup of TRIP

D.1.1 Network Architectures

Table D.1 The architecture of our ranker is adapted from the discriminator of RelGAN [Wu et al., 2019b]. LReLU represents the Leaky ReLU with a negative slop being 0.01. N_f, S_f, S_s, S_p and K denote the filter number, the filter size, the stride size, the padding size, and the attribute number, respectively.

Component	Input → Output Shape	Layer Information
Feature Layer	$2 \times (h, w, 3) \rightarrow 2 \times \left(\frac{h}{2}, \frac{w}{2}, 64\right)$	Conv-(N_f=64,S_f=4,S_s=2,S_p=1),LReLU
	$2 \times \left(\frac{h}{2}, \frac{w}{2}, 64\right) \rightarrow 2 \times \left(\frac{h}{4}, \frac{w}{4}, 128\right)$	Conv-(N_f=128,S_f=4,S_s=2,S_p=1),LReLU
	$2 \times \left(\frac{h}{4}, \frac{w}{4}, 128\right) \rightarrow 2 \times \left(\frac{h}{8}, \frac{w}{8}, 256\right)$	Conv-(N_f=256,S_f=4,S_s=2,S_p=1),LReLU
	$2 \times \left(\frac{h}{8}, \frac{w}{8}, 256\right) \rightarrow 2 \times \left(\frac{h}{16}, \frac{w}{16}, 512\right)$	Conv-(N_f=512,S_f=4,S_s=2,S_p=1),LReLU
	$2 \times \left(\frac{h}{16}, \frac{w}{16}, 512\right) \rightarrow 2 \times \left(\frac{h}{32}, \frac{w}{32}, 1024\right)$	Conv-(N_f=1024,S_f=4,S_s=2,S_p=1),LReLU
	$2 \times \left(\frac{h}{32}, \frac{w}{32}, 1024\right) \rightarrow 2 \times \left(\frac{h}{64}, \frac{w}{64}, 2048\right)$	Conv-(N_f=2048,S_f=4,S_s=2,S_p=1),LReLU
Rank Layer	$2 \times \left(\frac{h}{64}, \frac{w}{64}, 2048\right) \rightarrow \left(\frac{h}{64}, \frac{w}{64}, 2048\right)$	Subtract
	$\left(\frac{h}{64}, \frac{w}{64}, 2048\right) \rightarrow \left(\frac{h}{64}, \frac{w}{64}, 2048\right)$	Conv-(N_f=1,S_f=1,S_s=1,S_p=1),LReLU
	$\left(\frac{h}{64}, \frac{w}{64}, 2048\right) \rightarrow \left(\frac{h}{64} \times \frac{w}{64} \times 2048\right)$	Flatten
	$\left(\frac{h}{64} \times \frac{w}{64} \times 2048\right) \rightarrow (K,)$	Dense
GAN Layer	$\left(\frac{h}{64}, \frac{w}{64}, 2048\right) \rightarrow \left(\frac{h}{64} \times \frac{w}{64} \times 2048\right)$	Flatten
	$\left(\frac{h}{64} \times \frac{w}{64} \times 2048\right) \rightarrow (1,)$	Dense

D.1.2 Datasets

The collected RAs based on binary attributes contain fine-grained comparisons for the two face image datasets since their face images with same annotations may still have different

Table D.2 The architecture of our generator is quite similar to RelGAN's [Wu et al., 2019b]. The switchable normalization, denoted as SN, is applied to all layers excluding the last layer.

Component	Input → Output Shape	Layer Information
Down-sampling	$(h, w, 3+K) \rightarrow (h, w, 64)$	Conv-(N_f=64,S_f=7,S_s=1,S_p=3),SN,ReLU
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	Conv-(N_f=128,S_f=4,S_s=2,S_p=1),SN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Conv-(N_f=256,S_f=4,S_s=2,S_p=1),SN,ReLU
Residual Block (×6)	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Conv-(N_f=256,S_f=3,S_s=1,S_p=1),SN,ReLU
Up-sampling	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	Conv-(N_f=128,S_f=4,S_s=2,S_p=1),SN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	Conv-(N_f=64,S_f=4,S_s=2,S_p=1),SN,ReLU
	$(h, w, 64) \rightarrow (h, w, 3)$	Conv-(N_f=3,S_f=7,S_s=1,S_p=3),Tanh

strengths w.r.t. the specific attribute. In Fig. D.1a, $x_1 \succ x_2$ and $x_3 \succ x_4$ are two pairs of different degrees of comparisons. For the shoe dataset, their collected RAs based on same binary attributes contain pairwise comparisons with same degrees. This is because their images annotated with same annotations have same strengths w.r.t. the specific attribute. In Fig. D.1b, $x_1 \succ x_2$ and $x_3 \succ x_4$ are two pairs of same degrees of comparisons.

(a) "smile" attribute (b) "shoe & edge" attribute

Figure D.1 Relative attributes constructed based on given binary attributes.

D.1.3 Hyperparameter Settings

Standard Deviation of Structural Similarity (DSSIM) [1] Wu et al. [2019b] is used to set hyperparameters. Meanwhile, Accuracy of Attribute Swapping (AAS) Lample et al. [2017] is used to choose the optimal training epoch [2].

[1] DSSIM is to evaluate the performance of fine-grained translation.

[2] AAS is an auxiliary evaluation metric to avoid the trial case where the translation is failed but with low DDSIM, namley, the generator outputs the same image no matter what the latent variable v is.

Frechet Inception Distance (FID)/mean square error (MSE) [3] Saquil et al. [2018], Wu et al. [2019b], He et al. [2019] is not used for model selection. Because we found that TRIP equipped with the adversarial ranking paradigm can maintain a good-quality generation in a certain range of the hyperparameters.

[3] FID/MSE is to evaluate the quality of generation

References

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Yazeed Alharbi and Peter Wonka. Disentangled image generation through structured noise injection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5134–5142. IEEE, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017a.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017b.

Siddarth Asokan and Chandra Sekhar Seelamantula. Teaching a GAN what not to learn. In *Advances in Neural Information Processing Systems*, pages 3964–3975. Curran Associates, Inc., 2020.

Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on knowledge and data engineering*, 26(2):405–425, 2012.

Stephen D. Bay, Dennis F. Kibler, Michael J. Pazzani, and Padhraic Smyth. The UCI KDD archive of large data sets for data mining research and experimentation. *SIGKDD Explor.*, 2(2):81–85, 2000.

Monica Benito, Joel Parker, Quan Du, Junyuan Wu, Dong Xiang, Charles M Perou, and James Stephen Marron. Adjustment of systematic microarray data biases. *Bioinformatics*, 20(1):105–114, 2004.

Francisco Bernardo, Michael Zbyszynski, Rebecca Fiebrink, and Mick Grierson. Interactive machine learning for end-user innovation. In *AAAI Spring Symposia*, 2016.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2018.

Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Gustav Bredell, Kyriakos Flouris, Krishna Chaitanya, Ertunc Erdil, and Ender Konukoglu. Explicitly minimizing the blur error of variational autoencoders. In *International Conference on Learning Representations*, 2023.

Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. A stochastic treatment of learning to rank scoring functions. In *International Conference on Web Search and Data Mining*, pages 61–69. ACM, 2020.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *International Conference on Machine learning*, pages 89–96. ACM, 2005.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 186–193. ACM, 2006.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine Learning*. ACM, 2007a.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine Learning*, pages 129–136. ACM, 2007b.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Olivier Chapelle, Alekh Agarwal, Fabian H Sinz, and Bernhard Schölkopf. An analysis of inference with the universum. In *Advances in neural information processing systems*, pages 1369–1376. Curran Associates, Inc., 2008.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *International Conference on Web Search and Data Mining*, pages 193–202. ACM, 2013.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307. Curran Associates, Inc., 2017.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. doi: 10.1109/TPAMI.2023.3261988.

Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *Advances in Neural Information Processing Systems*, pages 6510–6520. Curran Associates, Inc., 2017.

Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5154–5163. IEEE, 2020.

Zheng Ding, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu. Guided variational autoencoder for disentanglement learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7920–7929. IEEE, 2020.

Chao Du, Kun Xu, Chongxuan Li, Jun Zhu, and Bo Zhang. Learning implicit generative models by teaching explicit ones. *arXiv preprint arXiv:1807.03870*, 2018.

Kevin Duh and Katrin Kirchhoff. Semi-supervised ranking for document retrieval. *Comput. Speech Lang.*, 25(2):261–281, 2011.

Jesse H. Engel, Matthew D. Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations*, 2018.

Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1000–1005. Springer, 2017.

Johann A Gagnon-Bartsch and Terence P Speed. Using control genes to correct for unwanted variation in microarray data. *Biostatistics*, 13(3):539–552, 2012.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680. Curran Associates, Inc., 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777. Curran Associates, Inc., 2017.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence*, pages 1753–1759. AAAI Press, 2017.

Anvita Gupta and James Zou. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1(2):105, 2019.

Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4941–4949. IEEE, 2017.

Pedro Antonio Gutiérrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2015.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

N. Görnitz, L. A. Lima, K. Müller, M. Kloft, and S. Nakajima. Support vector data descriptions and k-means clustering: One class? *IEEE transactions on neural networks and learning systems*, 29(9):3994–4006, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778. IEEE, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. IEEE, 2020.

Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019.

David Heckerman. A tutorial on learning with bayesian networks. *Innovations in Bayesian networks: Theory and applications*, pages 33–82, 2008.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637. Curran Associates, Inc., 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020.

Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Q Phung. MGAN: Training Generative Adversarial Nets with Multiple Generators. In *International Conference on Learning Representations*, 2018.

Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. How generative adversarial networks and their variants work: An overview. *ACM Comput. Surv.*, 52(1):10:1–10:43, 2019.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning*, pages 1558–1567. PMLR, 2017.

Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, 2008.

Jiabo Huang, Shaogang Gong, and Xiatian Zhu. Deep semantic clustering by partition confidence maximisation. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 8849–8858. IEEE, 2020.

Yating Huang and Hujun Yin. Manifold-enhanced cyclegan for facial expression synthesis. *The Imaging Science Journal*, 70(3):181–193, 2022.

Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

Aapo Hyvärinen, Jarmo Hurri, Patrik O Hoyer, Aapo Hyvärinen, Jarmo Hurri, and Patrik O Hoyer. Estimation of non-normalized statistical models. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, pages 419–426, 2009.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE conference on computer vision and pattern recognition*, pages 1125–1134. IEEE, 2017.

Arash Izadpanah and Richard L Gallo. Antimicrobial peptides. *Journal of the American Academy of Dermatology*, 52(3):381–390, 2005.

Laurent Jacob, Johann A Gagnon-Bartsch, and Terence P Speed. Correcting gene expression data when neither the unwanted variation nor the factor of interest are observed. *Biostatistics*, 17(1):16–28, 2016.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Harold Jeffreys, Bertha Jeffreys, and Bertha Swirles. *Methods of mathematical physics*. Cambridge university press, 1999.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *International Joint Conference on Artificial Intelligence*, pages 1965–1972. AAAI Press, 2017.

W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–127, 2007.

Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*, 2019.

Alexia Jolicoeur-Martineau. On relativistic f-divergences. In *International Conference on Machine Learning*, pages 4931–4939. PMLR, 2020.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2390–2399. PMLR, 2018.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems*, pages 12104–12114. Curran Associates, Inc., 2020a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119. IEEE, 2020b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2014.

Ruho Kondo, Keisuke Kawano, Satoshi Koide, and Takuro Kutsuna. Flow-based image-to-image translation with feature disentanglement. In *Advances in Neural Information Processing Systems*, pages 4168–4178. Curran Associates, Inc., 2019.

Brian Kulis and Michael I Jordan. Revisiting k-means: new algorithms via bayesian nonparametrics. In *International Conference on Machine Learning*, pages 1131–1138. Omnipress, 2012.

Gihyun Kwon and Jong Chul Ye. Diffusion-based image translation using disentangled style and content representation. In *International conference on learning representations*, 2023.

Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On controllable sparse alternatives to softmax. In *Advances in neural information processing systems*, pages 6423–6433. Curran Associates, Inc., 2018.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976. Curran Associates, Inc., 2017.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

Bo Li, Kaitao Xue, Bin Liu, and Yunyu Lai. Bbdm: Image-to-image translation with brownian bridge diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2023.

Chengtao Li, David Alvarez-Melis, Keyulu Xu, Stefanie Jegelka, and Suvrit Sra. Distributional Adversarial Networks. In *International Conference on Learning Representations*, 2018.

Peizhao Li, Han Zhao, and Hongfu Liu. Deep fair clustering for visual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9070–9079. IEEE, 2020a.

Xiao Li, Chenghua Lin, Chaozheng Wang, and Frank Guerin. Latent space factorisation and manipulation via matrix subspace projection. In *International Conference on Machine Learning*, pages 5916–5926. PMLR, 2020b.

Jianxin Lin, Zhibo Chen, Yingce Xia, Sen Liu, Tao Qin, and Jiebo Luo. Exploring explicit domain supervision for latent space disentanglement in unpaired image-to-image translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1254–1266, 2021.

Shili Lin. Rank aggregation methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):555–570, 2010.

Zinan Lin, Ashish Khetan, Giulia C. Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1505–1514. Curran Associates, Inc., 2018.

Jun Ling, Han Xue, Li Song, Shuhui Yang, Rong Xie, and Xiao Gu. Toward fine-grained facial expression manipulation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 37–53. Springer, 2020.

Jennifer Listgarten, Carl Kadie, Eric E Schadt, and David Heckerman. Correction for hidden confounders in the genetic analysis of gene expression. *Proceedings of the National Academy of Sciences*, 107(38):16465–16470, 2010.

Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. In *Advances in neural information processing systems*, pages 2590–2599. Curran Associates, Inc., 2018.

Rui Liu, Yu Liu, Xinyu Gong, Xiaogang Wang, and Hongsheng Li. Conditional adversarial generative flow for controllable image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7992–8001. IEEE, 2019.

Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3): 225–331, 2009.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Xiangyang Liu, Hongtao Lu, and Wenbin Li. Multi-manifold modeling for head pose estimation. In *IEEE International Conference on Image Processing*, pages 3277–3280. IEEE, 2010.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*, pages 3730–3738, 2015.

Tyler Lu and Craig Boutilier. Learning mallows models with pairwise preferences. In *International Conference on Machine Learning*, pages 145–152. Omnipress, 2011.

Ying Ma, Guoqiang Zhong, Yanan Wang, and Wen Liu. Metacgan: A novel gan model for generating high quality and diversity images with few training data. In *International Joint Conference on Neural Networks*, pages 1–7. IEEE, 2020.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *International Conference on Computer Vision*, pages 2794–2802. IEEE, 2017.

Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623. PMLR, 2016.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *International Conference on Learning Representations*, 2017.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *International Conference on Learning Representations*, 2018.

Daniel Moyer, Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. Invariant representations without adversarial training. In *Advances in Neural Information Processing Systems*, pages 9102–9111. Curran Associates, Inc., 2018.

Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, 2004.

Chuang Niu, Hongming Shan, and Ge Wang. Spice: Semantic pseudo-labeling for image clustering. *IEEE Transactions on Image Processing*, 31:7264–7278, 2022.

Donglin Niu, Jennifer G Dy, and Michael I Jordan. Iterative discovery of multiple alternativeclustering views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36 (7):1340–1353, 2013.

Augustus Odena. Semi-supervised learning with generative adversarial networks. *Workshop on Data-Efficient Machine Learning (ICML)*, 2016.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning*, pages 2642–2651. PMLR, 2017.

Utkarsh Ojha, Krishna Kumar Singh, Cho-Jui Hsieh, and Yong Jae Lee. Elastic-infogan: Unsupervised disentangled representation learning in class-imbalanced data. In *Advances in Neural Information Processing Systems*, pages 18063–18075. Curran Associates, Inc., 2020.

Yuangang Pan. *Robust Rank Aggregation and Its Applications*. PhD thesis, University of Technology Sydney, 2019.

Yuangang Pan and Ivor Tsang. Streamlining em into auto-encoder networks. In *OpenReview*, 2021. URL https://openreview.net/forum?id=EyDgK7q5vwJ.

Yuangang Pan, Bo Han, and Ivor W. Tsang. Stagewise learning for noisy k-ary preferences. *Machine Learning*, 107(8-10):1333–1361, 2018.

Yuangang Pan, Ivor W Tsang, Weijie Chen, Gang Niu, and Masashi Sugiyama. Fast and robust rank aggregation against model misspecification. *Journal of Machine Learning Research*, 23(23):1–35, 2022.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Devi Parikh and Kristen Grauman. Relative attributes. In *International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

David Keetae Park, Seungjoo Yoo, Hyojin Bahng, Jaegul Choo, and Noseong Park. MEGAN: Mixture of Experts of Generative Adversarial Networks for Multimodal Image Generation. In *International Joint Conference on Artificial Intelligence*, pages 878–884. AAAI Press, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 Workshop on Autodiff*, 2017.

Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *Proceedings of the European conference on computer vision (ECCV)*, pages 818–833, 2018.

Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7949–7958. IEEE, 2020.

Karthik Raman and Thorsten Joachims. Methods for ordinal peer grading. In *International Conference on Knowledge Discovery and Data Mining*, pages 1037–1046. ACM, 2014.

Michael Redmond and Alok Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678, 2002.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, pages 213–226. Springer, 2010.

Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

Tim Salimans, Ian J Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pages 2226–2234. Curran Associates, Inc., 2016.

Bidisha Samanta, Abir De, Gourhari Jana, Vicenç Gómez, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *Journal of Machine Learning Research*, 21(1):4556–4588, 2020.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

Yassir Saquil, Kwang In Kim, and Peter M. Hall. Ranking cgans: Subjective control over semantic image attributes. In *British Machine Vision Conference*, page 131. BMVA Press, 2018.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, volume 28, pages 3483–3491. Curran Associates, Inc., 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907. Curran Associates, Inc., 2019.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318. Curran Associates, Inc., 2017.

James Stewart, Daniel K Clegg, and Saleem Watson. *Multivariable calculus*. Cengage Learning, 2020.

Jinxuan Sun, Guoqiang Zhong, Yang Chen, Yongbin Liu, Tao Li, and Kaizhu Huang. Generative adversarial networks with mixture of t-distributions noise for diverse image generation. *Neural Networks*, 122:374–381, 2020.

David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54 (1):45–66, 2004.

Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. Mice: Mixture of contrastive experts for unsupervised image clustering. In *International conference on learning representations*, 2021.

Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6309–6318. Curran Associates, Inc., 2017.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (12), 2010.

Bernard L Welch. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Chieh Wu, Stratis Ioannidis, Mario Sznaier, Xiangyu Li, David Kaeli, and Jennifer Dy. Iterative spectral method for alternative clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 115–123. PMLR, 2018.

Chieh Wu, Jared Miller, Yale Chang, Mario Sznaier, and Jennifer Dy. Solving interpretable kernel dimensionality reduction. *Advances in Neural Information Processing Systems*, pages 7915–7925, 2019a.

Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y Chang, and Shih-Wei Liao. Relgan: Multi-domain image-to-image translation via relative attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5914–5922, 2019b.

Rongliang Wu and Shijian Lu. Leed: Label-free expression editing via disentanglement. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 781–798. Springer, 2020.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *International Conference on Machine Learning*, pages 1192–1199. ACM, 2008.

Yifan Xia, Wenbo Zheng, Yiming Wang, Hui Yu, Junyu Dong, and Fei-Yue Wang. Local and global perception generative adversarial network for facial expression synthesis. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1443–1452, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487. PMLR, 2016.

Min Yang, Chengming Li, Ying Shen, Qingyao Wu, Zhou Zhao, and Xiaojun Chen. Hierarchical human-like deep neural networks for abstractive text summarization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2744–2757, 2020.

Yinghua Yao, Yuangang Pan, Ivor W Tsang, and Xin Yao. Differential-critic gan: Generating what you want by a cue of preferences. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Lu Yi and Man-Wai Mak. Improving speech emotion recognition with adversarial data augmentation network. *IEEE Transactions on Neural Networks and Learning Systems*, 33 (1):1–13, 2020.

Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *IEEE conference on computer vision and pattern recognition*, pages 192–199. IEEE, 2014.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, pages 9422–9434, 2020.

Zac Yu and Adriana Kovashka. Syntharch: Interactive image search with attribute-conditioned synthesis. In *IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 645–654. IEEE, 2020.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. In *Advances in Neural Information Processing Systems*, pages 3609–3623. Curran Associates, Inc., 2022.

Zhibing Zhao, Peter Piech, and Lirong Xia. Learning mixtures of plackett-luce models. In *International Conference on Machine Learning*, pages 2906–2914. PMLR, 2016.

Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Learning to rank with ties. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–282. ACM, 2008.

Zixia Zhou, Yi Guo, and Yuanyuan Wang. Handheld ultrasound video high-quality reconstruction using a low-rank representation multipathway generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):575–588, 2021.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, pages 2223–2232. IEEE, 2017.