Trustworthy Machine Learning with Deep Generative Models

by

Dihong Jiang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2024

© Dihong Jiang 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Nidhi Hegde
Associate Professor, Computing Science
University of Alberta

Supervisors: Yaoliang Yu
Associate Professor, Computer Science
University of Waterloo

Sun Sun
Senior Research Officer
National Research Council, Canada

Internal Members: Pascal Poupart
Professor, Computer Science
University of Waterloo

Hongyang Zhang
Assistant Professor, Computer Science
University of Waterloo

Internal-External Member: Chengguo Weng
Professor, Statistics and Actuarial Science
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis is a presentation of my original research work summarized from three published papers, including two conference articles and one journal article as listed below. I am the *sole first author* of all three papers, where I mainly contributed ideas, implemented the proposed methods, and wrote papers. My co-authors are my supervisors, who provided me with invaluable help in brainstorming, refining ideas, and polishing my writing.

- D. Jiang, S. Sun, and Y. Yu (2022). "Revisiting flow generative models for out-of-distribution detection". In: *International Conference on Learning Representations*, which corresponds to Chapter 2.
- D. Jiang and S. Sun (2023). "DP-LFlow: Differentially private latent flow for scalable sensitive image generation". *Transactions on Machine Learning Research*, which corresponds to Chapter 3.
- D. Jiang, S. Sun, and Y. Yu (2023). "Functional Renyi Differential Privacy for Generative Modeling". *Advances in Neural Information Processing Systems*, vol. 36, which corresponds to Chapter 4.

Abstract

The recent decade has witnessed the remarkable progress of machine learning (ML) technologies across diverse domains, especially the deep generative models (DGMs) developed on high-dimensional data. However, as ML systems become integral to critical and sensitive applications, ensuring their trustworthiness becomes paramount beyond mere performance. Trustworthy machine learning encompasses a suite of principles and practices aimed at enhancing the reliability and safety of ML systems. This thesis investigates the intersection of trustworthiness and deep generative models, focusing on two pivotal aspects of trustworthiness: out-of-distribution (OOD) detection and privacy preservation.

Generative models serve purposes beyond generating realistic samples. Likelihood-based DGMs, e.g. flow generative models, can additionally compute the likelihood of input data, which can be used as an unsupervised OOD detector by thresholding the likelihood values. However, they have been found to occasionally assign higher likelihoods to OOD data compared to in-distribution (InD) data, raising concerns about their reliability and credibility in OOD detection. This thesis presents new insights demonstrating that flow generative models can reliably detect OOD data by leveraging their bijectivity property. The proposed approach involves comparing two high-dimensional distributions in the latent space. To manage the high dimensionality, we extend a univariate statistical test (e.g., Kolmogorov-Smirnov (KS) test) into higher dimensions using random projections. Our method indicates robustness across various datasets and data modalities.

The second focus of this thesis is on privacy preservation in DGMs. Generative models can also be seen as proxies for publishing training data, and there is growing interest in ensuring privacy preservation beyond generation fidelity in modern generative models. Differentially private generative models (DPGMs) offer a solution to protect individual data privacy in the training set of generative models. Existing methods either apply the workhorse algorithm in differentially private deep learning, e.g. differentially private stochastic gradient descent (DP-SGD), to DGMs, or use kernel methods by making the maximum mean discrepancy (MMD) objective differentially private. However, DP-SGD methods suffer from high training costs and poor scalability to stronger differential privacy (DP) guarantees, while kernel methods face the issue of mode collapse in generation.

We propose novel methods to sidestep the above challenges for both methods, respectively. To alleviate the training cost overhead and scalability issues on small privacy budgets for DP-SGD, we propose to train a flow generative model in a lower-dimensional latent space, which significantly reduces the model size and thereby avoids unnecessary computation in the full pixel space. This design is more resilient to larger noise perturbations, and enables us to be the first related work to present high-resolution image generations with DP constraints. On the other hand, to improve the model utility of MMD methods, we propose to make the MMD objective differentially private without truncating the reproducing kernel Hilbert Space (RKHS). To do so, we extend Rényi differential privacy (RDP) from vectors to functions, and then propose many useful building blocks to facilitate its

use in practice, e.g. Gaussian mechanism, composition theorems, and post-processing theorem. The proposed method consistently outperforms state-of-the-art methods by a large margin, especially under stronger DP guarantees.

The thesis is expected to provide new insights into the application of flow generative models in OOD detection, highlight practical challenges in training generative models with DP-SGD on high-dimensional datasets, bridge the gap between RDP and functional mechanisms, and expand the family of DPGM.

Acknowledgements

"*Destiny is always mysterious and ineffable for its unpredictability.*"

This is what I wrote when I completed my undergraduate study, and it feels also fitting to use it as a conclusion here now. Back then, I did not even expect myself to obtain a doctoral degree a few years later. Now here I am.

I heard a lot about what a PhD's life is like before my application, e.g. bitter-sweet, arduous, busy, or fulfilling. However, you will never have the real feeling until you are experiencing it by yourself. My first year coincided with the early stages of the pandemic, when everything went remote and online. That was the most isolating year I have ever had, with literally no social activities to my fellow students in Canada because I was studying at home in China. What is worse, most of my friends in China do not live or work in my hometown, so I did not have the chance to hang out with them either. But even if I had the chance I probably could not because the study and research were way more intensive than my expectation. I took four courses in my first term, because I really wished to complete my course requirements in the first year. Meanwhile, I need to get familiar with my new research topic and get my hands dirty to conduct my new PhD projects, which were quite different from what I did in my Master's research.

So my first and paramount thanks go to my supervisors, Yaoliang and Sun, who not only provided me with invaluable help in my research and life, but also showed impressive patience when I did not make too much progress towards the projects. I am also grateful for all their generous financial support in helping me attend conferences overseas. Yaoliang is especially good at abstract thinking and critical thinking, which I learned and benefited a lot from our weekly meetings. While progress is usually incremental, when I look back at what I have achieved combined, I start to realize how far I have gone, thanks to the guidance and support from my supervisors.

I would also like to thank all the professors on my examining committee—Nidhi, Pascal, Hongyang, and Chengguo—for your time, effort, and valuable suggestions on my thesis. I am sincerely grateful to each of you for agreeing to be part of my committee.

I had an unforgettable experience in seeking help from UWaterloo staffs. They always have extensive knowledge to answer my questions and make my study easier. Particularly, I cannot forget Paula, the previous CS PhD coordinator, spared her time and set up a video meeting with me in my first year to explain the intricate difference between Comp-I exam and PhD course requirement. Nadine, the current CS PhD coordinator, also has amazing knowledge about literally everything regarding CS PhD. She managed to answer all my questions expanding from courses, seminars, and finance to thesis defense. Yasmina, the graduate studies coordinator, is impressively warmhearted, responsive, and helpful, who did a wonderful job in coordinating everything relating to my thesis defense. I always get the most prompt email reply from you. Joe, who sends out the newsletter every week, is a funny and careful person. It is always delightful to communicate with you.

Next, I want to thank my labmates—Yiwei, Haoye, Jingjing, Zeou, Spencer, and Saber—for the academic discussions in the lab, from which I learned a great deal, and for the casual conversations and relaxing activities outside the lab. I also extend my thanks to friends who have or have not graduated from our university—Guojun, Ji, Kaiwen, Wanxin, Haotian, Dake and Shufan—for their support and help throughout my PhD journey and the fun times we shared off-campus.

Yufei, my roommate and one of my best friends in Fudan whom I can share any thoughts with, helped me a lot when I had a hard time in the middle of my PhD. I deeply appreciate all your encouraging words. Zeyu, an old friend during my Master's study, shared a lot with me about life and PhD study in the past few years. Our conversations were a huge relief to me and I hope they relieved you as well. Renjie, my another old friend from Fudan, whom I had not seen in almost six years, serendipitously met me in Hawaii last year when I was attending ICML 2023. We shared a lot in both old memories and life thoughts. I also want to thank Botao, Lingda, Dapeng, who are all my old friends in Fudan, for your help in my last year. I am increasingly more aware of how fortunate and precious it is to have friends all over the world.

My special thanks go to my previous supervisor at Fudan, Yu Ma, who is a wise, kindhearted and supportive man. He led me into the fascinating world of machine learning, and guided me to discover my true interests. He also encouraged me to pursue a PhD. I could not be where I am now if I did not meet him eight years ago.

Finally I want to thank my family, especially my parents and grandmother, for their firm support during my PhD study. The pandemic has separated us for years, and they had to endure the days without me while missing me dearly.

" 既然选择了远方，便只顾风雨兼程", one of my favorite verses from a contemporary Chinese poet, Guozhen Wang, which I have cherished since high school. I tried my best to get a satisfactory English translation with the help of ChatGPT (via a series of prompt engineering), which is the most poem-like verse I can get and I want to share it here:

"*For the far horizon,*

I shall go despite the wind and rain."

I hope that whenever I experience frustration in the future, the *wind and rain* I have endured during my PhD journey can remind me of how capable I am to get it over. I hope that my PhD experience will transform into my lifelong treasure and power me to advance to the *far horizon* till the end of the world, and I am sure it will.

Dedication

To my family.

Table of Contents

Examining Committee ii

Author's Declaration iii

Statement of Contributions iv

Abstract v

Acknowledgements vii

Dedication ix

List of Tables xv

List of Figures xviii

1 Introduction 1

1.1 Trustworthy Machine Learning . . . 2

1.1.1 Out-of-distribution (OOD) Detection . . . 2

1.1.2 Preserving Privacy of User Data . . . 2

1.1.2.1 Differential Privacy . . . 3

1.1.2.2 Rényi Differential Privacy (RDP): . . . 3

1.1.2.3 DP-SGD . . . 4

1.2 Generative Modeling . . . 5

1.2.1 Flow-based Generative Models . . . 6

1.2.2 Maximum Mean Discrepancy (MMD) . . . 6

1.2.3 Differentially Private Generative Model . . . 7

1.3 Outline and Contributions . . . 8

2 Out-of-distribution detection with flow generative models **9**
2.1 Preface . . . 9
2.2 Introduction . . . 10
2.3 OOD Detection with Flow Models . . . 12
2.3.1 OOD Detection: Group vs. Pointwise . . . 13
2.3.2 Flow-based Generative Models . . . 13
2.3.3 OOD Detection via Random Projections . . . 14
2.3.4 Improvement for Under-trained Flow Models . . . 17
2.4 Experimental Results . . . 18
2.4.1 Robustness of GOOD2KS . . . 18
2.4.2 Random Projection vs. Autoencoders . . . 20
2.4.3 Effect of Divergence Measures . . . 20
2.4.4 Generation Quality vs. OOD Detection . . . 21
2.4.5 Distribution Comparison in Latent vs. Sample Spaces . . . 23
2.5 Related Work . . . 24
2.6 Summary . . . 26

3 DP-LFlow: Differentially Private Latent Flow for Scalable Sensitive Image Generation **27**
3.1 Preface . . . 27
3.2 Introduction . . . 28
3.3 Preliminary . . . 30
3.3.1 Differential Privacy . . . 30
3.3.2 DP-SGD . . . 30
3.3.3 Flow-based Generative Models . . . 31
3.4 Method: DP-LFlow . . . 31
3.5 Experiments . . . 34
3.5.1 Experimental Setup . . . 34
3.5.2 Comparison with SoTA Baselines . . . 35
3.5.2.1 Under the Weak DP Guarantee $\epsilon = 10$ . . . 35
3.5.2.2 Under the Strong DP Guarantee $\epsilon = 1$ . . . 36

3.5.3 Generating High-resolution Images with DP Constraints 37
3.5.4 Differentially Private Out-of-distribution (OOD) Detection 38
3.5.5 Ablation Study 39
3.5.5.1 Flow in the Pixel Space vs. Latent Space 39
3.5.5.2 With vs. Without Partitioning the Dataset 39
3.6 Discussion 40
3.7 Related Work 43
3.8 Summary 44

4 Functional Rényi Differential Privacy for Generative Modeling 45
4.1 Preface 45
4.2 Introduction 46
4.3 Preliminary 47
4.3.1 Differential Privacy for Vectors (v-DP) 47
4.3.2 Rényi Differential Privacy for Vectors (v-RDP) 47
4.3.3 Gaussian Mechanism for v-DP and v-RDP 48
4.3.4 Differential Privacy for Functions (f-DP) 48
4.4 Rényi Differential Privacy for Functions (f-RDP) 50
4.4.1 Definition 50
4.4.2 Post-processing Theorem 50
4.4.3 Conversion to (ϵ, δ)-DP 51
4.4.4 Composition Theorems 51
4.4.5 Gaussian Mechanism 51
4.4.6 Subsampled Gaussian Mechanism (SGM) 52
4.5 An Application in DPGM 53
4.5.1 Background 53
4.5.2 Related Works 53
4.5.3 Methodology 54
4.5.4 Experiments 56
4.5.4.1 Conditional vs. Parallel 57
4.5.4.2 Grayscale Image Sets: MNIST and Fashion MNIST 57
4.5.4.3 RGB Image Sets: CelebA 57
4.6 Summary 58

5 Conclusion **61**

References **63**

APPENDICES **74**

A Appendix of Chapter 2 **75**

A.1 Datasets . . . 75
A.2 Implementation Details on Our Method . . . 76
A.3 Implementation Details of Benchmarks . . . 77
A.4 Framework . . . 77
A.5 Motivation for Groupwise Detection . . . 78
A.6 Point OOD Detection via Data Augmentation . . . 78
A.7 Additional Results . . . 80
A.7.1 GOOD Results using RealNVP - AUPR . . . 80
A.7.2 GOOD Results using Glow . . . 80
A.7.3 Comparison of Different Divergence Measures . . . 80
A.7.4 Batch Size = 20 . . . 81
A.7.5 Generation Quality vs. OOD Detection . . . 81
A.7.6 Intra-dataset OOD Detection . . . 84
A.7.7 OOD Detection on Time-series Data . . . 84
A.7.8 Investigation of CIFARs vs. LSUN . . . 85
A.7.9 Comparison with Average Density . . . 85

B Appendix of Chapter 3 **91**

B.1 Proof . . . 91
B.2 Datasets . . . 92
B.3 Rényi differential privacy (RDP) . . . 92
B.4 Framework . . . 92
B.5 Implementation . . . 93
B.5.1 Flow models . . . 93
B.5.2 Privacy implementation . . . 94
B.5.3 Fréchet Inception Distance (FID) . . . 94
B.5.4 Classification task . . . 94
B.5.5 Baselines . . . 95

C Appendix of Chapter 4 **96**

C.1 Proof 96

C.2 Datasets 99

C.3 Implementation 100

C.4 Additional Results 101

C.5 Broader Impact and Limitations 101

C.6 A Simplified View of RDP for Vectors and Functions 104

List of Tables

2.1 AUROC on RealNVP (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest AUROC are in boldface, and failure cases (where AUROC is below 0.5) are underlined. 19

2.2 Comparison in the sample and latent spaces. AUROCs are shown and segmented by |. S denotes detection in the sample space, while L-R and L-G denote detecting in the latent space using RealNVP and Glow, respectively. 23

2.3 Comparison in the sample and latent spaces with noise. AUROCs are shown and segmented by |. S denotes detection in the sample space, while L-R and L-G denote detecting in the latent space using RealNVP and Glow, respectively. 23

3.1 Quantitative comparison on MNIST and Fashion MNIST given $(10, 10^{-5})$-DP. Acc denotes classification accuracy, which is shown in %. $\uparrow$ and $\downarrow$ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021). Results of G-PATE and DataLens are cited from their papers, respectively. Ours* denotes our result without the post-processing trick. 36

3.2 Quantitative comparison on image datasets given $(1, 10^{-5})$-DP. FMNIST denotes Fashion MNIST. Acc denotes CNN classification accuracy, which is shown in %. $\uparrow$ and $\downarrow$ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of PATE-GAN and G-PATE are cited from Long et al. (2021). Results of DataLens are cited from B. Wang, Wu, et al. (2021). Ours* denotes our result without the post-processing trick. 39

3.3 Differentially private OOD detection results. The evaluation metric is shown by AUROC (higher is better, and 1.0 means InD and OOD likelihood are perfectly separable). 41

3.4 Ablation on partitioning the MNIST dataset. We compare the performance without post-processing tricks (smoothing and sharpening) for simplicity. . 41

4.1 Quantitative comparison on MNIST, Fashion MNIST (FMNIST) and CelebA. Acc denotes classification accuracy, which is shown in %. ↑ and ↓ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021) and Long et al. (2021). Results of PEARL and DPDM are cited from Dockhorn et al. (2023). Results of G-PATE and DataLens are cited from their papers, respectively. (*): DPDM did unconditional generation on CelebA. 60

A.1 AUROC on RealNVP for point OOD detection (higher is better). Our results are denoted by R1|R2, where R1 corresponds to GOOD1KS, and R2 corresponds to GOOD2KS. 80

A.2 AUPR on RealNVP (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest AUPR are in boldface, and failure cases (where AUPR is below 0.5) are underlined. 81

A.3 AUROC and AUPR on Glow (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest metrics are denoted in boldface, and failure cases (where metric is below 0.5) are underlined. 87

A.4 Comparison of KS test and CvM test on RealNVP. AUROC is shown (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. 88

A.5 Comparison with averaged density. Our results are denoted by GOOD1KS | GOOD2KS. We use the boldface for the highest metric, and underline the detection failure case (where metric is below 0.5). 88

A.6 AUROC and AUPR when batch size = 20. Our results are denoted by GOOD1KS | GOOD2KS. We use the boldface for the highest metrics, and underline the detection failure case (where metric is below 0.5). 89

A.7 Intra-dataset OOD detection using RealNVP. AUROC and AUPR are shown (higher is better). The batch size is fixed as 10. Our results are denoted by GOOD1KS|GOOD2KS. 90

A.8 OOD detection on EEG and ECG using RealNVP. AUROC and AUPR are shown (higher is better). The batch size is fixed as 10. Our results are denoted by GOOD1KS | GOOD2KS. 90

B.1 Network configurations for different datasets in the experiments. $\#h\_conv$ denotes the number of hidden sizes in the convolutional layers. $\#h\_lin$ denotes the number of hidden sizes in the linear layers of flow. $\#c$ denotes the length of latent code. $\#b$ means the number of blocks in flow. 93

C.1 Training parameters for retaining DP on MNIST and Fashion MNIST. Both variants on CelebA use parameters in the row of Conditional. q is the subsampling rate, and σ is the noise multiplier. 101

C.2 Quantitative comparison under $(10, 10^{-5})$. Acc denotes classification accuracy, which is shown in %. $\uparrow$ and $\downarrow$ refer to higher is better or lower is better, respectively. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021) and Long et al. (2021). Results of PEARL and DPDM are cited from Dockhorn et al. (2023). Results of G-PATE and DataLens are cited from their papers, respectively. (*): DPDM did unconditional generation on CelebA. 104

List of Figures

2.1 Histogram of Glow log-likelihoods. Glow assigns higher likelihood to OOD data. 11

2.2 Histogram of Glow log-likelihoods. Glow assigns lower likelihood to OOD data. 11

2.3 Examples of 2-D Gaussian in the latent space. In (a) and (b), left: the densities of InD and OOD; and middle and right: the density histograms of InD and OOD in each latent dimension. x means input variable while z denotes latent variable. 15

2.4 Random projections of 2-d Gaussians. As an example, we consider a standard Gaussian and correlated Gaussian that share the same marginal distributions. $proj_1$ is the projection along the axis $(1, 0)$, and $proj_2$ is the projection along $(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. For the standard Gaussian, both $proj_1$ (a) and $proj_2$ (b) follow $\mathcal{N}(0, 1)$; while for the correlated Gaussian, $proj_1$ (c) follows $\mathcal{N}(0, 1)$, but $proj_2$ (d) follows $\mathcal{N}(0, \frac{1}{5})$, instead of $\mathcal{N}(0, 1)$. 16

2.5 Random projections vs. autoencoder. (a)-(b) Performing random projection outperforms training an extra autoencoder in two OOD detection experiments. Besides, the OOD detection performance using RealNVP saturates as the number of random projections grows beyond 50. 20

2.6 Different divergences. KS and CvM test outperform JSD. A further comparison between KS and CvM is presented in Appendix A.7.3. 21

2.7 Model capacity vs. OOD detection. While the density histograms estimated from the simple Glow (a) and complex Glow (b) are similar, the generation from the simple Glow (c) and complex Glow (d) are noticeably different. More surprisingly, the OOD detection performance by simple Glow (AUROCs: 0.99|0.99) is better than the complex Glow (AUROCs: 0.96|0.95). . 22

3.1 FID vs. VAE size (in MB). We only vary the model complexity, with all the rest training parameters (e.g. subsampling rate, noise multiplier, training iterations) fixed. 32

3.2 Qualitative comparison on MNIST and Fashion MNIST under $(10, 10^{-5})$-DP. Images of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from (T. Cao et al. 2021). Images of G-PATE and DataLens are cited from their papers, respectively. 35

3.3 Qualitative comparison on MNIST and Fashion MNIST under $(1, 10^{-5})$-DP. Images of G-PATE and DataLens are cited from their papers, respectively. 37

3.4 Qualitative comparison on CelebA conditioned on gender under $(1, 10^{-5})$-DP. Images of DataLens are cited from the original paper. Top row: female. Bottom row: male. 38

3.5 Samples from DP-LFlow trained on CelebA-HQ under $(10, 10^{-5})$-DP. Top 2 rows: female. Bottom 2 rows: male. FID = 328.6, LR classification accuracy = 77.4. 40

3.6 Training a RealNVP (∼150 MegaBytes (MB)) on CelebA (a) non-privately with BN layers, (b) non-privately without BN layers, (c) under $(10, 10^{-5})$-DP without BN layers 42

4.1 Qualitative comparison under $(1, 10^{-5})$-DP on MNIST and Fashion MNIST 58

4.2 Qualitative comparison under $(0.2, 10^{-5})$-DP on MNIST and Fashion MNIST 59

4.3 The qualitative comparison on CelebA. $\delta = 10^{-5}$. DPDM is unconditional. 59

A.1 The framework of our method. 78

A.2 An example of distribution based group anomaly. (a) Two 2-d Gaussians with similar density and large overlap in support. (b) Density histograms largely overlap, where likelihood score only achieves a detection AUROC of 0.57. 79

A.3 Model capacity vs. OOD detection. Images are generated from RealNVP trained on CelebA. Assume CIFAR-10 as OOD, and our GOOD1KS and GOOD2KS results are segmented by |. (a) Generated images from a RealNVP with 1 block and 2048 hidden channels. AUROCs = 0.98|0.99. (b) Generated images from a RealNVP with 16 blocks and 512 hidden channels. AUROCs = 0.88|0.88. 82

A.4 Training epochs vs. OOD detection. Images are generated from RealNVP trained on CelebA. Assume CIFAR-10 as OOD, and our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 10 epochs. AUROCs = 0.78|0.78. (b) Trained for 100 epochs. AUROCs = 0.81|0.81. (c) Trained for 150 epochs. AUROCs = 0.86|0.86. (d) Trained for 200 epochs. AUROCs = 0.88|0.88. 83

A.5 Training epochs vs. OOD detection. Images are generated from a Glow with $K = 16, L = 3, h = 128$ trained on CIFAR-10. Treating SVHN as OOD, our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 5 epochs. AUROCs = 0.96|0.96. (b) Trained for 10 epochs. AUROCs = 0.96|0.95. (c) Trained for 100 epochs. AUROCs = 0.76|0.84. (d) Trained for 200 epochs. AUROCs = 0.87|0.84. 83

A.6 Training epochs vs. OOD detection. Images are generated from a Glow with $K = 16, L = 2, h = 128$ trained on FMNIST. Treating MNIST as OOD, our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 5 epochs. AUROCs = 1.00|1.00. (b) Trained for 10 epochs. AUROCs = 1.00|1.00. (c) Trained for 100 epochs. AUROCs = 0.90|1.00. (d) Trained for 200 epochs. AUROCs = 0.92|1.00. 84

A.7 Per-dimensional mean and variance comparison. (a)-(b) Per-dimensional mean and variance of CIFARs and LSUN are close, which implies that CIFARs and LSUN share a similar distribution of normalized pixel values. (c)-(d) As a comparison, CIFARs vs SVHN have more clearly separable per-dimensional variance. 86

B.1 The framework of DP-LFlow 93

C.1 Qualitative comparison under $(10, 10^{-5})$-DP on MNIST and Fashion MNIST 102

C.2 Qualitative comparison under $(10, 10^{-5})$-DP on CelebA 103

Chapter 1

Introduction

The rapid advancement of computational resources and the increasing accessibility to large-scale datasets have brought transformative changes in machine learning (ML) especially deep learning (DL) across various domains in recent decades, such as computer vision (Voulodimos et al. 2018), speech recognition (Nassif et al. 2019), and economics (Charpentier et al. 2021). Central to this evolution is deep generative models (DGMs), which have demonstrated remarkable capabilities in generating realistic data, modeling complex distributions, and enhancing various machine learning tasks. Some high-tech companies have already productized a few real-world applications and achieved great success. Significant examples include ChatGPT and DALLE from OpenAI, Llama from Meta, and Stable Diffusion from Stability AI.

Despite their success, the deployment of deep generative models in real-world applications faces significant challenges related to trustworthiness. According to Varshney (2022), trustworthy machine learning contains a spectrum of critical aspects, including data privacy, distribution shift, fairness, robustness, interpretability, ethics, etc. These dimensions are essential to ensure that machine learning systems can be reliably and safely integrated into sensitive environments, i.e. there is increasing demand to care about more than just synthesizing realistic samples for deep generative models.

In this thesis, we make some initial attempts to address part of these challenges by exploring methods to handle two aspects of trustworthiness, i.e. distribution shift and privacy, of deep generative models, which we hope can contribute to bridging the gap between the powerful capabilities of deep generative models and the stringent requirements for their safe and reliable use in practical applications.

The following sections in this chapter will include a more detailed introduction of trustworthy machine learning (particularly the two notions we focus on), various generative modeling techniques, and an outline combined with a summarized contribution of this thesis.

1.1 Trustworthy Machine Learning

In this thesis, I will focus on two concerns on the notion of trustworthiness, i.e., distribution shift (aka out-of-distribution) and privacy.

1.1.1 Out-of-distribution (OOD) Detection

ML algorithms are usually built upon the assumption that training and test data are identically distributed (in-distribution), which, however, becomes almost unrealistic in real life. On one hand, training and test data are usually collected differently, which may lead to a minor shift in distribution. In this case, ML model with high cross validated accuracy may perform poorly on test data (Varshney 2022). On the other hand, sometimes test data are mistakenly drawn from a completely different distribution, but a well-trained classification model will still output an in-distribution label with high confidence, which is detrimental to real-world decision making, especially for security-sensitive applications such as medical diagnosis or autonomous driving (Emmott et al. 2015). For example, Nitsch et al. (2021) show that a neural network that is trained to classify pedestrians, cars, truck, and cyclists may recognize a deer as a car with high confidence, and the high confidence will undermine the trustworthiness of the system and force the autonomous vehicles to (unnecessarily) take more conservative actions.

Therefore, OOD detection aims to identify and mitigate those risks by flagging or rejecting anomalous (OOD) inputs, which is essential for enhancing the reliability and security of ML systems.

Based on the label availability, OOD detection methods can be roughly categorized as follows:

Supervised vs. Unsupervised: By assigning labels to both In-Distribution (InD) and OOD data instances, supervised OOD detection methods mainly involve training a (binary) classifier to distinguish InD and OOD inputs (Chalapathy and Chawla 2019), which, however, suffer from three main drawbacks: (1) there is limited label availability since it is hard and expensive to obtain abundant labeled data in practice; (2) OOD is usually unpredictable and not well-defined. The detector trained on existing OOD data may not be able to detect OOD data from an unseen class; (3) the common class imbalance issue will make the classification more challenging. In contrast, unsupervised OOD detection methods entirely rely on the intrinsic patterns of data themselves, which are more convenient and generalized, thus we adopt unsupervised methods in this thesis.

1.1.2 Preserving Privacy of User Data

While the access to large-scale datasets results in great success in modern ML systems, there are increasing concerns on privacy leakage during the use of the data, especially the

sensitive data (e.g. face images, medical records) that can be exploited by a malicious party, even though the original ML applications never intentionally do so. For example, Fredrikson et al. (2015) show that an attacker, who is only given a name and the white-box access to a face recognition model, can successfully recover face images of a particular person who appears in the training set (which is also known as the model inversion attack).

Prior efforts on developing privacy-preserving techniques include naive data anonymization (Narayanan and Shmatikov 2008), k-anonymity (Sweeney 2002), l-diversity (Machanavajjhala et al. 2007), t-closeness (N. Li, T. Li, et al. 2007), semantic security (Goldwasser and Micali 1984), information-theoretic privacy (Diaz et al. 2019), and differential privacy (DP) (Dwork 2006), where the last one is recognized as a rigorous quantization of privacy, and becomes the gold-standard in the current ML community.

1.1.2.1 Differential Privacy

Differential privacy upper bounds the deviation in the output distribution of a randomized algorithm given incremental deviation in the input. Formally, we have the following definition:

Definition 1 ((ϵ, δ)-DP (Dwork, Roth, et al. 2014))**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $D, D' \in \mathcal{D}$ and for any measurable subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that*

$$Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta$$

where adjacent inputs (a.k.a. neighbouring datasets) D and D' only differ in one entry. Particularly, when $\delta = 0$, we say that $\mathcal{M}$ is ϵ-DP.

A famous theorem, i.e. post-processing theorem, which is utilized by existing works (as well as ours) for proving DP guarantee of a published model, is given by:

Theorem 1 (Post-processing theorem, (Dwork, Roth, et al. 2014))**.** *If $\mathcal{M}$ satisfies (ϵ, δ)-DP, $F \circ \mathcal{M}$ will satisfy (ϵ, δ)-DP for any function F, where $\circ$ denotes the composition operator.*

Post-processing theorem states that any function that is independent of the training dataset preserves differential privacy. If we have a differentially private generative model (DPGM) (which will be widely discussed in the following chapters), then sampling from the model will not incur further privacy cost.

1.1.2.2 Rényi Differential Privacy (RDP):

Rényi differential privacy (RDP) extends ordinary DP by the use of α-Rényi divergence (Rényi 1961), which provides tighter and easier composition property than the ordinary DP notion. Formally, we have:

Definition 2 ((α, ϵ)-RDP (Mironov 2017)). *A randomised mechanism $\mathcal{M}$ is (α, ϵ)-RDP if for all adjacent inputs D, D', Rényi's α-divergence (of order $\alpha > 1$) between the distribution of $\mathcal{M}(D)$ and $\mathcal{M}(D')$ satisfies:*

$$\mathbb{D}_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) := \frac{1}{\alpha - 1} \log \mathbb{E}_{Z \sim Q} \left(\frac{p(Z)}{q(Z)} \right)^\alpha \leq \epsilon,$$

where p and q are the density of $\mathcal{M}(D)$ and $\mathcal{M}(D')$, respectively (w.r.t. some dominating measure μ).

Conveniently, RDP is linearly composable:

Theorem 2 (Composition of RDP (Mironov 2017)). *If mechanism $\mathcal{M}_i$ satisfies (α, ϵ_i)-RDP for $i = 1, 2, \ldots, k$, then the composed mechanism $\mathcal{M}_1 \circ \ldots \circ \mathcal{M}_k$ satisfies $(\alpha, \sum_{i=1}^k \epsilon_i)$-RDP.*

Gaussian mechanism is widely adopted for achieving RDP:

Definition 3 (Gaussian mechanism for RDP (Dwork, Roth, et al. 2014; Mironov 2017)). *Let $f : \mathcal{D} \to \mathbb{R}^d$ be an arbitrary d-dimensional function with sensitivity:*

$$\Delta_2 f = \max_{D, D'} \| f(D) - f(D') \|_2$$

for all adjacent datasets $D, D' \in \mathcal{D}$. The Gaussian mechanism $\mathcal{M}_\sigma$ perturb the output of f with Gaussian noise:

$$\mathcal{M}_\sigma = f(D) + \mathcal{N}(0, \sigma^2 \cdot \mathbb{I})$$

where $\mathbb{I}$ is the identity matrix. Then, $\mathcal{M}_\sigma$ satisfies $(\alpha, \frac{\alpha (\Delta_2 f)^2}{2\sigma^2})$-RDP.

1.1.2.3 DP-SGD

While DP is initially designed to limit information leakage of individual data when sharing aggregate statistics of a database, it is later extended to deep learning by Abadi et al. (2016), where the authors proposed Differentially Private Stochastic Gradient Descent (DP-SGD) that offers a framework for training a deep learning model with DP guarantees.

DP-SGD applies Gaussian mechanisms to the gradients that are used to update the model weights. To do so, it makes two notable extensions to the ordinary SGD: (1) clipping the L_2 norm of individual gradients, to ensure the L_2 sensitivity of the gradient is upper bounded; (2) adding calibrated Gaussian noise to the aggregate of the gradients. The model weight is iteratively updated with the privatized gradients, and the final DP guarantee is retained by sequentially composing the Gaussian mechanisms with DP (RDP) guarantees.

However, DP-SGD has two practical challenges: (1) the individual gradient clipping introduces heavy computation time overhead, which makes it inefficient especially when

training large models; (2) it does not scale well at stronger DP guarantees (which means more noise perturbations), which we will elaborate more in Chapter 3. Looking for a more efficient application of DP-SGD or a more scalable alternative is still of interest to both researchers and practitioners.

Besides, there is always a *privacy-utility tradeoff* when applying mechanisms with DP guarantees, e.g. running DP-SGD. Intuitively, the individual data privacy is preserved at the cost of losing model accuracy, due to the introduced noise. The more noise we add, the more privacy we preserved, but the model utility is worse. Improving the *privacy-utility tradeoff* is constantly an essential objective in related works.

1.2 Generative Modeling

Generative modeling is a fundamental and rapidly evolving area of machine learning that focuses on modeling and understanding the underlying data distribution to generate new data samples that resemble the original dataset, which has wide-ranging applications, from creating realistic images/text/video to enhancing data augmentation and anomaly detection.

Generative models aim to learn the underlying density $p(\mathbf{x})$ from a given dataset. Once the distribution is learned, the model can generate new samples $\mathbf{x} \sim p(\mathbf{x})$. Thanks to the development of deep learning, an emerging family of generative models, i.e. deep generative models, prevail in generative modeling, where generative models are parameterized by deep neural networks. Some prominent DGMs include generative adversarial network (GAN) (Goodfellow et al. 2014), variational autoencoder (VAE) (Kingma and Welling 2013), normalizing flow (Rezende and Mohamed 2015), auto-regressive models (Van den Oord et al. 2016), and diffusion models (Sohl-Dickstein et al. 2015), which can synthesize realistic samples that even humans could hardly tell the difference.

Among these DGMs, likelihood-based DGMs, e.g. normalizing flow, can provide an additional benefit of density estimation, thus benefiting the applications when data density or likelihood is of interest, e.g. assessing data property (e.g. tail, skewness), presenting the distribution, detecting anomaly input (Pimentel et al. 2014; J. Wang et al. 2019), summarizing Bayesian posteriors (Givens and Hoeting 2012), information regularization (Szummer and Jaakkola 2002), and open set recognition (Herbei and Wegkamp 2006).

However, with the rise of privacy concerns, how to protect individual data privacy when training a generative model has become another factor of consideration other than generation quality, especially when sensitive data are involved. One possible solution is to develop a *differentially private generative model*, which can synthesize data that are distributionally similar to the training data while satisfying DP guarantees. Currently, there are two mainstream approaches to do so: (1) running DP-SGD with an existing DGM, and (2) making the Maximum Mean Discrepancy (MMD) objective in kernel methods differentially private.

In this section, I will first summarize one representative likelihood-based DGM, i.e. flow-based generative model, then briefly introduce another generative modeling technique via kernel methods. How both generative modeling methods can be extended with DP guarantees will be included in the end.

1.2.1 Flow-based Generative Models

A flow-based generative model simply learns a transformation T, typically a diffeomorphism, that pushes a latent distribution p_0 (e.g., Gaussian or uniform) to the in-distribution p, i.e.

$$Z \sim p_0 \implies \mathsf{T}(Z) \approx p, \quad \text{also denoted as } p \approx \mathsf{T}_{\#} p_0.$$

In particular, when T is diffeomorphic, we have the familiar change-of-variable formula:

$$p_{\mathsf{T}}(\mathbf{x}) = p_0(\mathbf{z}) / |\mathsf{T}'(\mathbf{z})| = p_0\left(\mathsf{T}^{-1}(\mathbf{x})\right) / \left|\mathsf{T}'\left(\mathsf{T}^{-1}(\mathbf{x})\right)\right|,$$

where $|\mathsf{T}'(\mathbf{z})|$ denotes the absolute value of the determinant of the Jacobian of T. A flow model parameterizes T through a neural network and estimates T by minimizing some divergence between p_{T} (the model density) and p (the data density), based on the training data $\mathcal{D}$. Once T is learned, we can easily generate new data by simply sampling Z from p_0 and then pushing through T, i.e. $X = \mathsf{T}(Z)$.

For example, using Kullback–Leibler (KL) divergence to measure the fitness, the objective becomes:

$$\mathbb{D}_{KL}(p(\mathbf{x}) || p_{\mathsf{T}}(\mathbf{x})) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{p_{\mathsf{T}}(\mathbf{x})} \, \mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[-\log p_{\mathsf{T}}(\mathbf{x})] - \mathbb{H}[p]$$

As the entropy of true distribution p is a fixed term, training the flow generative model amounts to maximizing the likelihood of observed data.

1.2.2 Maximum Mean Discrepancy (MMD)

Assuming a feature map $\phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) of some kernel, MMD (Gretton et al. 2012) is a non-parametric distance measure that compares two distributions p, q by:

$$\mathcal{L}_{\mathrm{MMD}^2}(p, q) = \|\mathbb{E}_{\mathbf{x} \sim p}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{w} \sim q}[\phi(\mathbf{w})]\|_{\mathcal{H}}^2,$$

where $\mathbb{E}_{\mathbf{x} \sim p}[\phi(\mathbf{x})] \in \mathcal{H}$ is also known as the (kernel) mean embedding (KME) of p. Note that RKHS is an *infinite-dimensional space*.

Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that $k(\mathbf{x}, \mathbf{w}) = \langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle_{\mathcal{H}}$, we can play the kernel trick to compute the (squared) MMD in an alternative way:

$$\mathcal{L}_{\text{MMD}^2}(p, q) = \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim p} k(\mathbf{x}, \mathbf{x}') - 2\mathbb{E}_{\mathbf{x} \sim p, \mathbf{w} \sim q} k(\mathbf{x}, \mathbf{w}) + \mathbb{E}_{\mathbf{w},\mathbf{w}' \sim q} k(\mathbf{w}, \mathbf{w}')$$

Sriperumbudur et al. (2011) suggest that if k is a characteristic kernel (e.g., Gaussian kernel), then MMD $= 0$ iff $p = q$, which makes MMD a practical tool in many applications, such as two-sample test (Gretton et al. 2012) and generative modeling (e.g., C.-L. Li et al. 2017; Y. Li et al. 2015).

1.2.3 Differentially Private Generative Model

Differentially private generative models emerge as a solution to protect individual privacy of the data that are used to train generative models. Most DPGMs are developed on either tabular (low dimensional, usually below 20 dimensions) or image (high dimensional, around or above 1000 dimensions) datasets, where image datasets are considered more challenging due to the larger input dimension, which is also the target input domain in this thesis.

Existing works fall into either of the following two categories:

- **DP-SGD:** running DP-SGD for a deep generative model, e.g. GAN (D. Chen et al. 2020; Torkzadehmahani et al. 2019; Xie et al. 2018), VAE (Acs et al. 2018; Q. Chen et al. 2018; Jiang, G. Zhang, et al. 2022; Weggenmann et al. 2022), normalizing flow (Lee et al. 2022; Waites and Cummings 2021), diffusion (Dockhorn et al. 2023; Ghalebikesabi et al. 2023). Beyond naively running DP-SGD for the whole model, there are some careful designs to minimize the noise perturbation to the training while retaining the same level of DP guarantee, such as only adding noise to part of the model (that will be publicly released) (D. Chen et al. 2020; Jiang, G. Zhang, et al. 2022). This line of work is subject to low training efficiency and poor scalability to stronger DP guarantees.
- **MMD:** running kernel methods via MMD (Harder et al. 2021; Liew et al. 2022; Vinaroz et al. 2022), where the noise is added to the (empirical) kernel mean embedding of the training data, which can be viewed as a privatized mean query of the entire training set with individual privacy preserved. Thereafter, the training objective reduces to matching the kernel mean embedding of generated data and the privatized kernel mean embedding of the real data. Since the KME in theory is in an infinite-dimensional space, which is intractable to inject multivariate Gaussian noise, prior researchers propose to approximate the KME by a finite-dimensional feature embedding. This line of method tends to generate 'mean-like' images, thus restricting its use in downstream ML tasks. More details can be found in Chapter 4.

1.3 Outline and Contributions

Each of the remaining chapters is centered around a full-length article where I am the sole first author. The main contributions are summarized as follows:

In Chapter 2,

- We propose a robust group-wise OOD detection method using flow generative models based on a goodness of fit test and random projection.
- We propose both one-sample and two-sample version of our methods based on the training sufficiency, where the two-sample version is assumption-free.
- We confirm on larger models and real datasets that model accuracy may be negatively correlated with OOD detection performance.

In Chapter 3,

- We explore the practical challenges of training a DP flow via DP-SGD on image sets, and we propose an efficient solution by reducing the training from the full pixel space to a lower-dimensional latent space.
- The proposed method indicates scalability on small privacy budget settings and large images. Uniquely, it is the first related work to present high-resolution (up to 256×256) image synthesis with DP constraints.

In Chapter 4,

- We generalize RDP from vectors to functional outputs, and develop all building blocks to facilitate its use in practice.
- We propose a novel approach for training a DPGM, by making the loss function in the RKHS private without truncating the RKHS, thus maximally preserving the learning capability of a Gaussian kernel.
- The proposed method consistently outperforms other SOTA kernel- and non-kernel-based methods by a large margin, especially when epsilon is small (better privacy-utility tradeoff).

Chapter 2

Out-of-distribution detection with flow generative models

2.1 Preface

Typical machine learning classifiers may predict a label among training classes for a test sample with high confidence, even when the test sample comes from a distribution that is completely different from the training data, which may lead to severe consequences in real-world applications such as clinical diagnosis systems. As suggested in (Bishop 1994), one intuitive solution is to train a density estimation model (usually deep generative models of which the likelihood is tractable, e.g. flow-based model, auto-regressive model and variational autoencoder (VAE)) to approximate the training distribution, or in-distribution (InD), and then identify the test sample with low density as OOD since the likelihood of InD is maximized in training. However, Nalisnick, Matsukawa, Teh, Gorur, et al. (2019) found that deep generative models may assign higher likelihood to OOD input than InD input. In this work, we show that flow generative models can reliably detect OOD input by utilizing its bijectivity. The proposed method is shown to be robust in various datasets or data modalities.

Abstract: Deep generative models have been widely used in practical applications such as the detection of out-of-distribution (OOD) data. In this work, we aim to re-examine the potential of generative flow models in OOD detection. We first propose a simple combination of univariate one-sample statistical test (e.g., Kolmogorov-Smirnov) and random projections in the latent space of flow models to perform OOD detection. Then, we propose a two-sample version of our test to account for imperfect flow models. Quite distinctly, our method does not pose parametric assumptions on OOD data and is capable of exploiting any flow model. Experimentally, firstly we confirm the efficacy of our method against state-of-the-art baselines through extensive experiments on several image

datasets; secondly we investigate the relationship between model accuracy (e.g., the generation quality) and the OOD detection performance, and found surprisingly that they are not always positively correlated; and thirdly we show that detection in the latent space of flow models generally outperforms detection in the sample space across various OOD datasets, hence highlighting the benefits of training a flow model. Our code is available at: `https://github.com/dihjiang/OOD-flow`.

The following sections are adapted from the main paper and appendix of D. Jiang, S. Sun, and Y. Yu (2022). "Revisiting flow generative models for out-of-distribution detection". In: *International Conference on Learning Representations*.

2.2 Introduction

Reliably detecting out-of-distribution (OOD) data, also known as anomaly, plays an important role in many deployed machine learning systems, particularly so when the system may be under attack by a malicious adversary (Chalapathy and Chawla 2019; Markou and Singh 2003a,b; Toth and Chawla 2018). The bulk of existing OOD detection algorithms, in one way or another, boils down to thresholding the (density) likelihood, which can be conveniently estimated by modern flow-based generative models. However, quite surprisingly, recent studies (Nalisnick, Matsukawa, Teh, Gorur, et al. 2019) reveal that flows that are trained to maximize the likelihood of in-distribution (InD) data may actually assign a higher likelihood to OOD data. For example, the flow-based model Glow (Kingma and Dhariwal 2018) trained on CIFAR-10 that contains natural images (e.g., dog, cat, and ship) is found to assign a higher likelihood to SVHN that consists of house numbers. The reason behind is still an open question. Prior researchers ascribe this phenomenon to some confounding factors in input data or trained models, such as the background statistics (Ren et al. 2019), the inductive biases (Kirichenko et al. 2020), the input complexity (Serrà et al. 2020), and the model accuracy (Choi et al. 2018). Such surprising and counter-intuitive observations lead to natural questions on the applicability of flow models for performing the OOD detection by merely thresholding the log-likelihood.

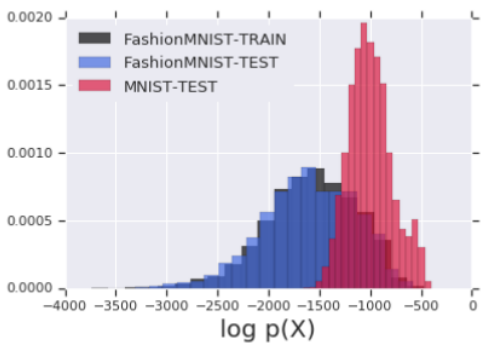
It is worth mentioning that this behavior is not symmetric (see Figure 2.1 and Figure 2.2 below, which are cited from Figure 2 and 6 in Nalisnick, Matsukawa, Teh, Gorur, et al. (2019)), here we show that the asymmetry is not surprising by briefly proving that Figure 2.1 implies Figure 2.2.

Let p, q denote two distributions, i.e., InD and OOD, respectively. Figure 2.1 shows that the log-likelihood histogram of OOD data is centered at a larger value than InD data. In particular,

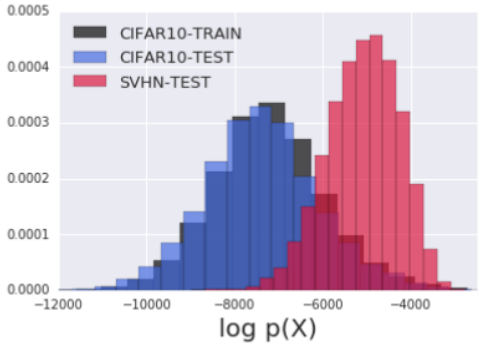
$$\mathbb{E}_{\mathsf{X}\sim q} \log p(\mathsf{X}) > \mathbb{E}_{\mathsf{X}\sim p} \log p(\mathsf{X}), \tag{2.1}$$

namely that on average the log-likelihood (trained on InD p) is larger for OOD data than InD data. On the other hand, Figure 2.2 shows that

$$\mathbb{E}_{\mathsf{X}\sim q} \log q(\mathsf{X}) > \mathbb{E}_{\mathsf{X}\sim p} \log q(\mathsf{X}), \tag{2.2}$$

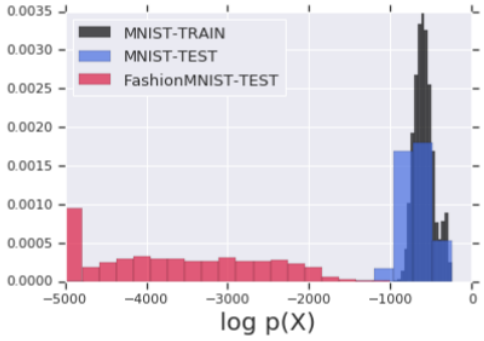


(a) Training on FashionMNIST, testing on MNIST

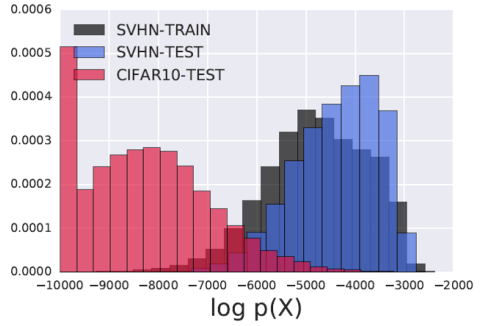


(b) Training on CIFAR-10, testing on SVHN

Figure 2.1: Histogram of Glow log-likelihoods. Glow assigns higher likelihood to OOD data.



(a) Training on MNIST, testing on FashionMNIST



(b) Training on SVHN, testing on CIFAR-10

Figure 2.2: Histogram of Glow log-likelihoods. Glow assigns lower likelihood to OOD data.

i.e., if we swap the roles of p and q, on average the log-likelihood (trained on "InD" q) is larger for "InD" data than "OOD" data. This surprising result can be explained from the following inequality:

$$\begin{aligned} 0 &\leq \mathbb{D}_{KL}(q||p) + \mathbb{D}_{KL}(p||q) \\ &= \left[\mathbb{E}_{\mathsf{X}\sim p} \log p(\mathsf{X}) - \mathbb{E}_{\mathsf{X}\sim q} \log p(\mathsf{X})\right] + \left[\mathbb{E}_{\mathsf{X}\sim q} \log q(\mathsf{X}) - \mathbb{E}_{\mathsf{X}\sim p} \log q(\mathsf{X})\right], \end{aligned}$$

which means Eq. 2.2 must hold when Eq. 2.1 holds, i.e., Figure 2.1 implies Figure 2.2. In other words, when likelihood (as an OOD detector) fails in one certain InD and OOD setting, it will succeed if we switch the definition of InD and OOD for the same two distributions (but the reverse is not necessarily true).

In this work, we re-examine the ability of flow-based generative models for OOD detection. Flow models typically transform a prior distribution that is easy to sample from to the data distribution via an invertible mapping (Dinh et al. 2017; Kingma and Dhariwal 2018; Rezende and Mohamed 2015; Tabak and Vanden-Eijnden 2010). Building on

this property, we propose to compare the training and test samples in the latent space of a flow model, as opposed to relying on point-wise scoring functions such as the log-likelihood. To this end, we divide test samples into minibatches (groups). Examples of group OOD detection with generative models include for instance Chalapathy, Toth, et al. (2018), Nalisnick, Matsukawa, Teh, and Lakshminarayanan (2019), J. Song et al. (2019), and Y. Zhang et al. (2020), most of which consider either the raw input or certain representation of the raw input in the sample space for detection. In contrast, we exploit data representations in the latent space, and confirm its advantage experimentally. Moreover, to cope with the curse of dimensionality (as is typical in image datasets), we propose to leverage random projections (Friedman et al. 1984), which frees one from designing any extra network architectures and is computationally very efficient. Our proposed detection algorithm combines classic univariate statistical tests and random projections in the latent space of flow models.

Besides the scoring function used for detection and the representation of data, model accuracy also greatly affects OOD detection performance (Choi et al. 2018; L. Zhang et al. 2021). To account for under-trained flow models, we propose a two-sample version of our detection algorithm for practical use. Surprisingly, in our experiments, we confirm that model accuracy, indicated by its generation quality, may not always be positively correlated with its OOD detection performance.

We summarize our contributions as follows:

- We propose the OOD detection algorithms GOOD1KS and GOOD2KS for ideal and under-trained flow models, respectively, which pose no parametric assumption on the OOD data. To evade the curse of dimensionality, we propose to randomly project the latent space of flow models onto the real line and perform univariate statistical tests (such as Kolmogorov-Smirnov, KS) there. Our method is computationally very efficient, requires no extra network architecture, and unifies OOD detection in both sample space and latent space.

- Experimentally, 1) we compare with the state-of-the-art benchmarks on various image datasets and obtain competitive results; 2) we confirm on larger models and real datasets that model accuracy may not always be positively correlated with OOD detection performance; and 3) we compare detection in the sample space versus in the latent space of the flow model, and reveal the superiority and robustness of the latter, hence highlighting the potential of flow models in OOD detection.

2.3 OOD Detection with Flow Models

In this section, we state the OOD detection problem and propose statistical tests that exploit modern flow generative models (either well- or under-trained).

2.3.1 OOD Detection: Group vs. Pointwise

Let $\mathcal{D} = \{X_1, \ldots, X_n\} \overset{\text{i.i.d.}}{\sim} p$ be a sample from an in-distribution (InD) p. Our goal is to construct a statistical test that can decide if a test sample $\{Y_1, \ldots, Y_m\}$ is from p (InD) or some unknown out-of-distribution (OOD) q. When $m = 1$, i.e. we examine one test sample at a time, it is often called the pointwise OOD detection (POOD) while group OOD detection (GOOD) refers to $m > 1$. According to Xiong et al. (2011), group OOD can be characterized as (1) point based, where each individual point in the group is anomalous, and (2) distribution based, where a group of points shows a different pattern while any single point may seem regular. Therefore, distribution based group anomaly is only detectable by GOOD methods. Practical applications include detecting Higgs Bosons as a group of collision events in high-energy particle physics (Muandet and Schölkopf 2013), and detecting distributed denial-of-services attacks via a group of multi-sensor networks (X. Chen and S. Yu 2016). We construct a synthetic Gaussian experiment in Appendix A.5 to illustrate this point. Thus in this work, we will focus on studying the applicability of modern flow generative models to the group OOD detection problem.

2.3.2 Flow-based Generative Models

A flow-based generative model simply learns a transformation T, typically a diffeomorphism, that pushes a latent distribution p_0 (e.g., Gaussian or uniform) to the in-distribution p, i.e.

$$Z \sim p_0 \implies \mathsf{T}(Z) \approx p, \quad \text{also denoted as } p \approx \mathsf{T}_{\#} p_0.$$

In particular, when T is diffeomorphic, we have the familiar change-of-variable formula:

$$p_{\mathsf{T}}(\mathbf{x}) = p_0(\mathbf{z}) / |\mathsf{T}'(\mathbf{z})| = p_0\left(\mathsf{T}^{-1}(\mathbf{x})\right) / \left|\mathsf{T}'\left(\mathsf{T}^{-1}(\mathbf{x})\right)\right|,$$

where $|\mathsf{T}'(\mathbf{z})|$ denotes the absolute value of the determinant of the Jacobian of T. A flow model parameterizes T through a neural network and estimates T by minimizing some divergence between p_{T} (the model density) and p (the data density), based on the training data $\mathcal{D}$. Once T is learned, we can easily generate new data by simply sampling Z from p_0 and then pushing through T, i.e. $X = \mathsf{T}(Z)$.

Denote $X_1 \overset{\text{d}}{=} X_2$ if they follow the same distribution. For any transformation T, it is clear that

$$X_1 \overset{\text{d}}{=} X_2 \implies \mathsf{T}(X_1) \overset{\text{d}}{=} \mathsf{T}(X_2),$$

and the converse also holds if T is invertible. In fact, a sufficiently regular transformation T may preserve many familiar statistical divergences D, i.e.

$$\mathsf{D}(X_1, X_2) = \mathsf{D}(\mathsf{T}(X_1), \mathsf{T}(X_2)),$$

for instance, when D is the f-divergence, such as the Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence, T is diffeomorphic, and X_1 and X_2 are continuous (Csiszár 1963). Another example is the Kolmogorov-Smirnov (KS) distance for real-valued X_i and monotonic T:

$$\mathsf{D}(X_1, X_2) := \sup_x |F_1(x) - F_2(x)|, \tag{2.3}$$

where F_i is the CDF of X_i, as well as the Cramér-von Mises (CvM) divergence (Darling 1957):

$$\mathsf{D}(X_1, X_2) := \int [F_1(x) - F_2(x)]^2 \mathrm{d}F_2(x).$$

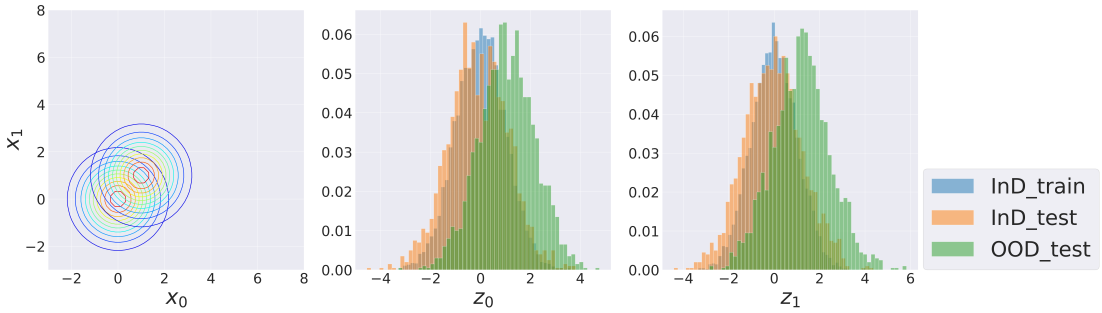
Thus, for group OOD detection, we can first train an invertible flow model based on the InD samples and then construct a scoring function or statistical test either in the sample space (where the test sample $\{Y_j\}$ resides), or in the latent space (where $\{\mathsf{T}^{-1}(Y_j)\}$ resides). However, since both Y_j and its pre-image $\mathsf{T}^{-1}(Y_j)$ are typically of high dimension and the sample size m is comparatively small, some compromise needs to be made in order to evade the curse of dimensionality, which gets even more problematic for flow models whose latent dimension is greater than input dimension (J. Chen et al. 2020; Grcić et al. 2021). For instance, Y. Zhang et al. (2020) fit a multivariate Gaussian in the latent space using $\{\mathsf{T}^{-1}(Y_j)\}$ and compute analytically the KL divergence between two Gaussians. Nalisnick, Matsukawa, Teh, and Lakshminarayanan (2019) project Y_i onto the real line using the estimated log-likelihood function $\log p_{\mathsf{T}}$ and construct a typicality test there, while Choi et al. (2018) instead employ the Watanabe-Akaike Information Criterion (WAIC).

To illustrate the above idea, we compare in Figure 2.3 the distributions of $\mathsf{T}^{-1}(\mathbf{X}_{\text{train}})$ and $\mathsf{T}^{-1}(\mathbf{X}_{\text{test}})$ in the latent space using synthetic 2-D Gaussian datasets and the flow model RealNVP (Dinh et al. 2017). We observe that in the latent space the distribution of the OOD samples (i.e., $\mathsf{T}^{-1}(\mathbf{X}_{\text{OOD-test}})$) is distinct (more so in Fig 1(b) than Fig 1(a)) from the distribution of the InD samples (i.e., both $\mathsf{T}^{-1}(\mathbf{X}_{\text{InD-train}})$ and $\mathsf{T}^{-1}(\mathbf{X}_{\text{InD-test}})$). Moreover, we found that the transformation T^{-1} can roughly keep the statistical distance in the latent space. In other words, distributions that are far away from the in-distribution in the sample space also tend to remain far away in the latent space.

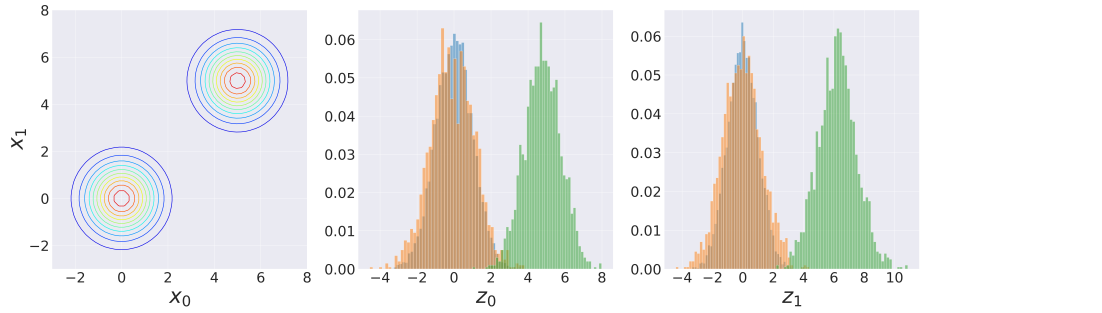
2.3.3 OOD Detection via Random Projections

For a well-trained flow model, its inverse transformation T^{-1} should bring the in-distribution samples to follow (approximately) the prior distribution p_0, such as the commonly used standard normal. It is thus natural to perform OOD detection by comparing $\mathsf{T}^{-1}(\mathbf{X}_{\text{test}})$ against the prior distribution p_0, whereas the InD training samples $\mathbf{X}_{\text{train}}$ are used to train the flow parameterized by T. To accommodate any prior distribution p_0, we propose to extend standard statistical tests such as the KS distance in Eq. 2.3 to high dimensions.

To our best knowledge, multivariate KS test has only been studied sporadically in the literature, due to the computational difficulty in enumerating the maximum in high



(a) InD: $\mathcal{N}([0,0],\mathbf{I})$ and OOD: $\mathcal{N}([1,1],\mathbf{I})$



(b) InD: $\mathcal{N}([0,0],\mathbf{I})$ and OOD: $\mathcal{N}([5,5],\mathbf{I})$

Figure 2.3: Examples of 2-D Gaussian in the latent space. In (a) and (b), left: the densities of InD and OOD; and middle and right: the density histograms of InD and OOD in each latent dimension. x means input variable while z denotes latent variable.

dimensions. For instance, Justel et al. (1997) proposed a complicated numerical procedure to compute the KS distance (incidentally also in the latent space) but only for bivariate distributions. Instead, we propose to randomly project high-dimensional random vectors to the real line and run classic KS test there. We note that similar ideas have been used to train generative models (e.g. Bonneel et al. 2015; Friedman et al. 1984; Kolouri et al. 2016; Liutkus et al. 2019; Nguyen et al. 2021; Paty and Cuturi 2019), while we allow any flow models and use random projection only to construct the KS test (and related).

Figure 2.4 provides a simple example to illustrate the main idea. Consider two 2-d Gaussian distributions: $\mathcal{N}([0,0],\mathbf{I})$ and $\mathcal{N}([0,0],[1,0.8;0.8,1])$. Obviously, for the standard 2-d Gaussian all of its random projections along normalized directions follow $\mathcal{N}(0,1)$, whereas for the shown correlated Gaussian, only the projections along the coordinate axes follow $\mathcal{N}(0,1)$. Therefore, we can distinguish these two distributions by comparing their projections along random directions.

More generally, the following theorem provides the theoretical basis for distributional comparison using random projections.

Theorem 3 (Cuesta-Albertos et al. (2007)). *Let X and Y be two $\mathbb{R}^d$-valued random vectors. Suppose the absolute moments $m_k := \mathbb{E}\|X\|^k$ are finite and $\sum_{k=1}^{\infty} (m_k)^{-1/k} = \infty$. If*

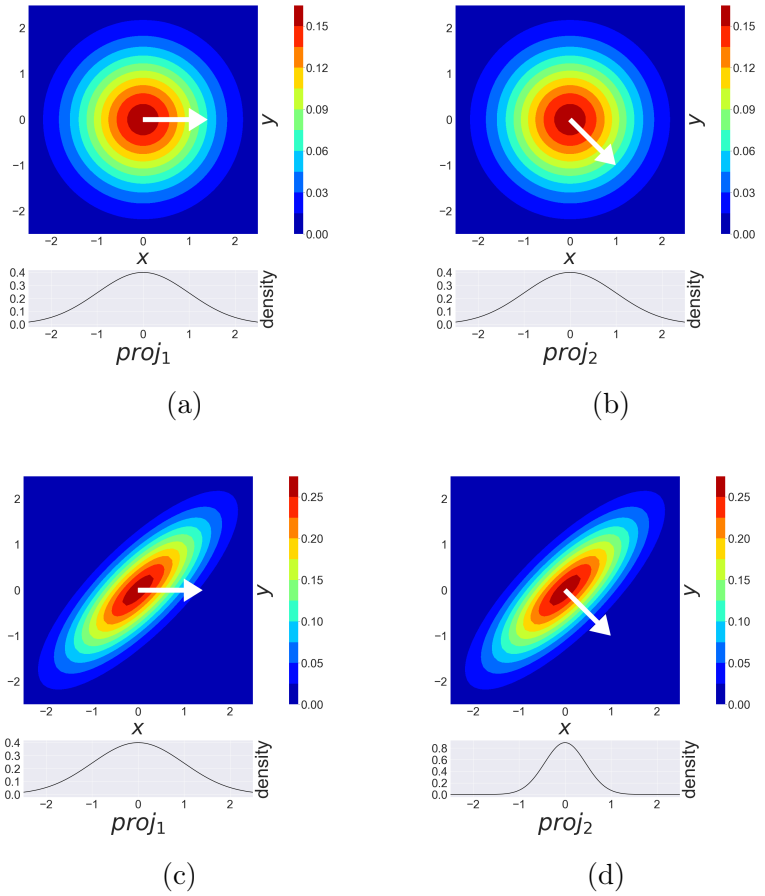


Figure 2.4: Random projections of 2-d Gaussians. As an example, we consider a standard Gaussian and correlated Gaussian that share the same marginal distributions. $proj_1$ is the projection along the axis $(1, 0)$, and $proj_2$ is the projection along $(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$. For the standard Gaussian, both $proj_1$ (a) and $proj_2$ (b) follow $\mathcal{N}(0, 1)$; while for the correlated Gaussian, $proj_1$ (c) follows $\mathcal{N}(0, 1)$, but $proj_2$ (d) follows $\mathcal{N}(0, \frac{1}{5})$, instead of $\mathcal{N}(0, 1)$.

the set $W = \{\mathbf{w} \in \mathbb{R}^d : \mathbf{w}^\top X \overset{\mathrm{d}}{=} \mathbf{w}^\top Y\}$ *has positive Lebesgue measure, then* $X \overset{\mathrm{d}}{=} Y$.

The assumption $\sum_k (m_k)^{-1/k} = \infty$, known as Carleman's condition, is very mild: it is satisfied if the underlying moment generating function is finite around the origin (hence such distributions are uniquely determined by their moments). Most distributions used in practice, such as the Gaussian distribution, clearly satisfy Carleman's condition. Put differently, Theorem 3 implies that an "enough bulk" of random directions allows us to distinguish the projections of X and Y; see Figure 2.4 for an illustration.

To be more specific, given a limited collection of test samples $\mathbf{X}_{\text{test}}$ and a flow model T, we first transform into the latent space and obtain $\mathbf{Z} = \mathsf{T}^{-1}(\mathbf{X}_{\text{test}})$. Then, we sample n (uniformly) random directions $\mathbf{W} \in \mathbb{R}^{d \times n}$, which are obtained by normalizing i.i.d. samples

from the d-dimensional standard Gaussian. When the flow model T is well-trained, the inverse transformation T^{-1} applied to InD samples will bring them to follow approximately the latent prior distribution, e.g., d-dimensional standard Gaussian. As a result, projection onto each random direction yields close proximity to the univariate standard Gaussian, which classic statistical tests such as KS would be able to pick up. In practice, we found that averaging over different random directions leads to slightly better and more robust performance, although the benefits quickly saturate as we increase the number of random projections. Crucially, the KS distance in Eq. 2.3, with F_1 being the empirical distribution of random projections $W^\top \mathbf{Z}$ and F_2 the latent prior distribution, can be computed in linear time by just enumerating x over each projected sample.

We call the resulting algorithm group OOD detection based on one-sample KS test (GOOD1KS) and summarize it in Algorithm 1. Its computation requires one pass of the (inverse) flow model and the remaining operations are linear-time. A higher value of the KS statistics k_{ij} indicates a lower similarity between the test sample and the latent prior distribution, which can be taken as a metric of OOD-ness. We note that our algorithm is completely general and efficient:

- unlike Y. Zhang et al. (2020) we do not require any matrix inversion or determinant and we avoid the difficult problem of estimating high dimensional covariance matrices when only very limited test samples are available;
- in principle, we can work with any latent prior distribution and any univariate statistical tests. As pointed out by Jaini et al. (2020), a heavier tailed latent distribution, or even discrete ones, than the standard Gaussian may be advantageous in certain settings. Similarly, other statistical tests may prove useful if we desire to zoom in certain parts of the distribution. Our choice of the KS test is motivated by our experimental settings below and serves as a concrete example.
- our algorithm can work with any flow model T. For instance, we may even take $\mathsf{T} = \mathrm{Id}$, in which case Algorithm 1 reduces to performing statistical tests in the sample space. Thus, our algorithm unifies the two perspectives: test in the sample space vs. test in the latent space, which we will compare experimentally below.

2.3.4 Improvement for Under-trained Flow Models

When the flow model is not well-trained (perhaps even intentionally, for instance, if we take $\mathsf{T} = \mathrm{Id}$), the effectiveness of the one-sample test in Algorithm 1 becomes questionable even for InD samples. A simple fix is then to run the two-sample version of the KS test in Eq. 2.3, where F_1 is the empirical distribution of the projected test samples while F_2 is now the empirical distribution of the InD training samples. More concretely, in Algorithm 1 we additionally derive $\mathbf{Z}_{\text{train}} = \mathsf{T}^{-1}(\mathbf{X}_{\text{train}})$ and project similarly to obtain $\mathbf{S}_{\text{train}} = \mathbf{W}^\top \mathbf{Z}_{\text{train}}$. Then, in Line 5 we substitute the latent prior distribution (e.g. standard Gaussian) with

Algorithm 1: Group OOD detection based on one-sample KS test (GOOD1KS).

Input: Test OOD samples $\mathbf{X}_{\text{test}}$ splitted into m groups $\mathbf{X}_1, \cdots, \mathbf{X}_m$ with each $\mathbf{X}_i \in \mathbb{R}^{d\times b}$ (b for batch size and d for dimension); random projection matrix $\mathbf{W} \in \mathbb{R}^{d\times n}$.

for $i \leftarrow 1$ ***to*** m **do**
  $\mathbf{Z}_i = \mathsf{T}^{-1}(\mathbf{X}_i) \in \mathbb{R}^{d\times b}$ `// transform into the latent space`
  $\mathbf{S}^{(i)} = \mathbf{W}^\top \mathbf{Z}_i \in \mathbb{R}^{n\times b}$ `// project onto` n `random directions`
  for $j \leftarrow 1$ ***to*** n **do**
    $k_{ij} = \mathrm{KS}(\mathbf{S}^{(i)}_{j:}, \mathcal{N}(0,1))$ `// conduct one-sample KS test`
  $k_i \leftarrow \frac{1}{n}\sum_{j=1}^{n} k_{ij}$ `// average over` n `random directions`
compute AUROC for $\mathbf{X}_{\text{test}}$ based on all k_i's

the empirical distribution of the j-the row of $\mathbf{S}_{\text{train}}$. Below we call this modification as group OOD detection based on two-sample KS test (GOOD2KS). We note that the computational complexity of GOOD2KS remains similar to Algorithm 1, and it is equally flexible: we can now even take snapshots of T obtained during training the flow model, and run GOOD2KS on all of them. The detailed algorithm for GOOD2KS is presented in Appendix A.2.

2.4 Experimental Results

We perform extensive experiments to compare our proposed OOD detection algorithms with the state-of-the-art (SOTA) group OOD detection benchmarks, i.e. Typicality test (TyTest) (Nalisnick, Matsukawa, Teh, and Lakshminarayanan 2019) and the KL divergence based Out-of-Distribution Detection (KLOD) (Y. Zhang et al. 2020). Implementation details of benchmarks are given in Appendix A.3. We test on two popular flow models: Glow (Kingma and Dhariwal 2018) and RealNVP (Dinh et al. 2017) (see Appendix A.2 for more details about network architecture). For evaluation, we focus on the Area Under Receiver Operating Characteristic (AUROC/AUC) and Area Under Precison-Recall curve (AUPR), which are commonly used in OOD detection. We compare the OOD detection performance across a wide variety of image datasets, including grayscale and RGB image datasets (please refer to Appendix A.1 for more details).

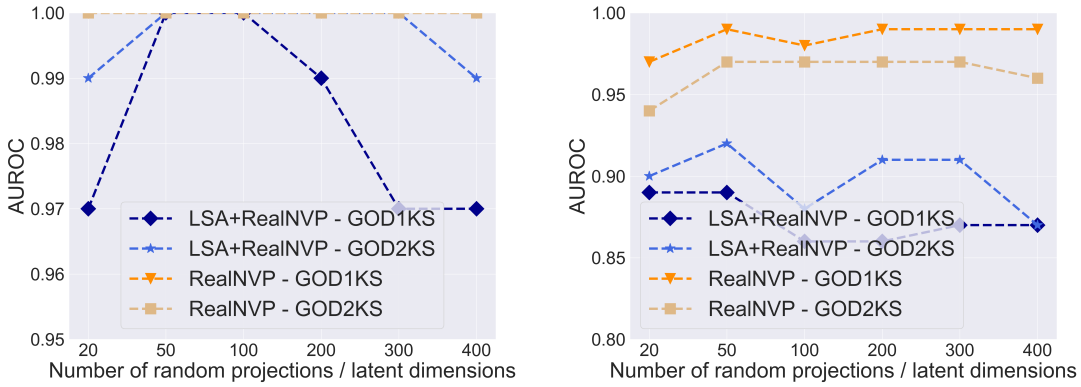
2.4.1 Robustness of GOOD2KS

In Table 2.1 we compare the performance of GOOD1KS and GOOD2KS with the flow model RealNVP against the benchmark algorithms. For fair comparison, we adopt the same RealNVP model in the benchmarks. AUPR comparison is given in Table A.2 in Appendix A.7.1. We use InD to denote the in-distribution dataset and OOD to denote

Table 2.1: AUROC on RealNVP (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest AUROC are in boldface, and failure cases (where AUROC is below 0.5) are underlined.

InD	OOD	batch size = 5			batch size = 10		
		TyTest	KLOD	Ours	TyTest	KLOD	Ours
FMNIST	MNIST	0.77	0.95	**0.99**|**0.99**	0.82	0.96	**1.00**|**1.00**
	KMNIST	0.97	0.97	**0.99**|**0.99**	0.97	0.98	**1.00**|**1.00**
	Omniglot	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
CIFAR-10	SVHN	**0.89**	0.29	0.88|0.83	0.95	0.35	**0.99**|0.98
	CelebA	0.67	**0.90**	0.76|0.77	0.81	**0.99**	0.89|0.90
	LSUN	0.60	**0.72**	0.61|0.62	0.70	**0.81**	0.66|0.70
CIFAR-100	SVHN	0.92	0.23	0.89|0.84	0.98	0.29	**0.99**|0.98
	CelebA	0.42	**0.83**	0.74|0.74	0.46	**0.97**	0.88|0.88
	LSUN	0.49	**0.63**	0.57|0.59	0.53	**0.72**	0.61|0.65
SVHN	CIFAR-10	**1.00**	0.99	0.89|0.93	**1.00**	0.99	0.95|0.98
	CIFAR-100	**1.00**	0.99	0.90|0.93	**1.00**	**1.00**	0.96|0.98
	CelebA	**1.00**	**1.00**	0.92|0.94	**1.00**	**1.00**	0.98|0.99
	LSUN	**1.00**	**1.00**	0.93|0.94	**1.00**	**1.00**	0.98|0.99
CelebA	CIFAR-10	0.98	**0.99**	0.92|0.93	**1.00**	**1.00**	0.98|0.99
	CIFAR-100	0.98	**0.99**	0.91|0.92	**1.00**	**1.00**	0.98|0.99
	SVHN	0.78	0.80	**0.97**|0.96	0.81	0.99	**1.00**|**1.00**
	LSUN	**1.00**	**1.00**	0.91|0.93	**1.00**	**1.00**	0.98|0.99

different test datasets. To implement group OOD detection, we divide the test samples into small batches and vary the batch size for comparison. We use 50 random projections in all cases, and set it as the default for all experiments on RealNVP. As expected, the performance improves with the batch size (see Appendix A.7.4 for results with batch size 20). We observe that, the performance of the two benchmarks and our GOOD1KS on these image datasets is unstable, and for some datasets the detection performance can be rather poor (i.e., AUROC below 0.5). For example, for TyTest with CIFAR-100 (InD) and CelebA (OOD), the AUROCs are 0.42 and 0.46; for KLOD with CIFAR-10 (InD) and SVHN (OOD), the AUROCs are 0.29 and 0.35. In contrast, the performance of our GOOD2KS is more robust and is generally satisfactory over all datasets. Results on Glow is shown in Table A.3 in Appendix A.7.2. Again, our GOOD2KS exhibits robustness over all datasets. Benchmarks also discuss detecting CIFAR-100 as OOD when training a model on CIFAR-10, and their AUROCs are around random guess. We did the same experiment and obtain similar results. The result is not surprising, since CIFAR-10 and CIFAR-100 are similar datasets, which highlights the need to quantify OOD-ness.



(a) FMNIST/MNIST (b) CIFAR-10/SVHN

Figure 2.5: Random projections vs. autoencoder. (a)-(b) Performing random projection outperforms training an extra autoencoder in two OOD detection experiments. Besides, the OOD detection performance using RealNVP saturates as the number of random projections grows beyond 50.

2.4.2 Random Projection vs. Autoencoders

When the number of random projections is less than the input dimension, we essentially perform dimensionality reduction before comparing the distributions. Therefore, it is natural to compare with other dimensionality reduction methods such as autoencoders. To construct the benchmark, we first feed the input image to an autoencoder and then use the latent code from the encoder as the input in our GOOD1KS/GOOD2KS. The only difference is that with autoencoder we now skip the step of random projections. As an example, we use the Latent Space Autoregression (LSA) (Abati et al. 2019) as the autoencoder. Figure 2.5 shows the comparison results on FMNIST/MNIST (InD/OOD) and CIFAR-10/SVHN using RealNVP. The batch size is fixed as 10. We vary the number of random projections or the latent dimension from 20 to 400, and found that using random projections outperforms autoencoder with the same level of dimensionality reduction. It is possible to improve the detection performance with an autoencoder by using a more complex network than LSA. However, random projection is still appealing, as it works effectively and requires no extra networks.

2.4.3 Effect of Divergence Measures

We empirically compare the KS test with the CvM test and the JS divergence (JSD). Both the KS and the CvM tests are non-parametric, while JSD requires the density to be estimated. We use the following heuristic approach to estimate the empirical density: dividing the latent values of the test batch into 20 bins, and use the normalized count in each bin as the empirical density (note that we can skip this step in both KS and CvM).

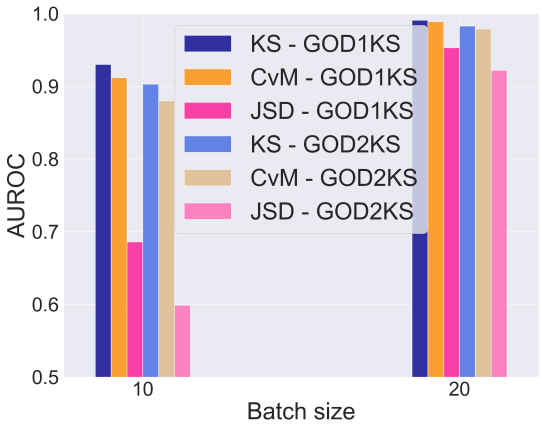


Figure 2.6: Different divergences. KS and CvM test outperform JSD. A further comparison between KS and CvM is presented in Appendix A.7.3.

Therefore, the computation time for calculating JSD is much longer. For implementation, we use the Scipy.stats library in Python to computation these measures. As an example, in Figure 2.6 we show results on one challenging dataset pair CIFAR-10/SVHN (InD/OOD) using RealNVP (results on Glow are similar). We consider two batch sizes: 10 and 20. The number of random projections is set to be 10 for all experiments. We can observe that JSD is outperformed by the other two in all cases, and the performance of KS and CvM is comparable. We further compare KS and CvM on more dataset pairs for different batch sizes (see Appendix A.7.3). The observation is generally consistent with KS being slightly superior to CvM, especially for the smaller batch size.

2.4.4 Generation Quality vs. OOD Detection

Intuitively, a more accurate flow model is expected to lead to better OOD detection performance. In this section, we investigate the relationship between the model accuracy and the OOD detection performance, where we measure a model's accuracy by its ability to generate visually realistic high-quality new images. To impose different levels of model accuracy we consider two factors: model capacity and training time. We fix the batch size to 10 in this experiment.

Model capacity: Consider Glow, where K denotes the number of steps of flow in each block, L denotes the number of blocks, and h denotes the number of hidden channels. For comparison we trained two different Glow models on CIFAR-10: (1) a simple Glow with $K = 3, L = 3, h = 64$, and (2) a complex Glow with $K = 16, L = 3, h = 128$. Figure 2.7 shows the comparison of the log-likelihood histograms and the generated images. We can see that while the histograms of the two log-likelihood are similar, the generation quality is noticeably different with the complex model generating much better images. We then

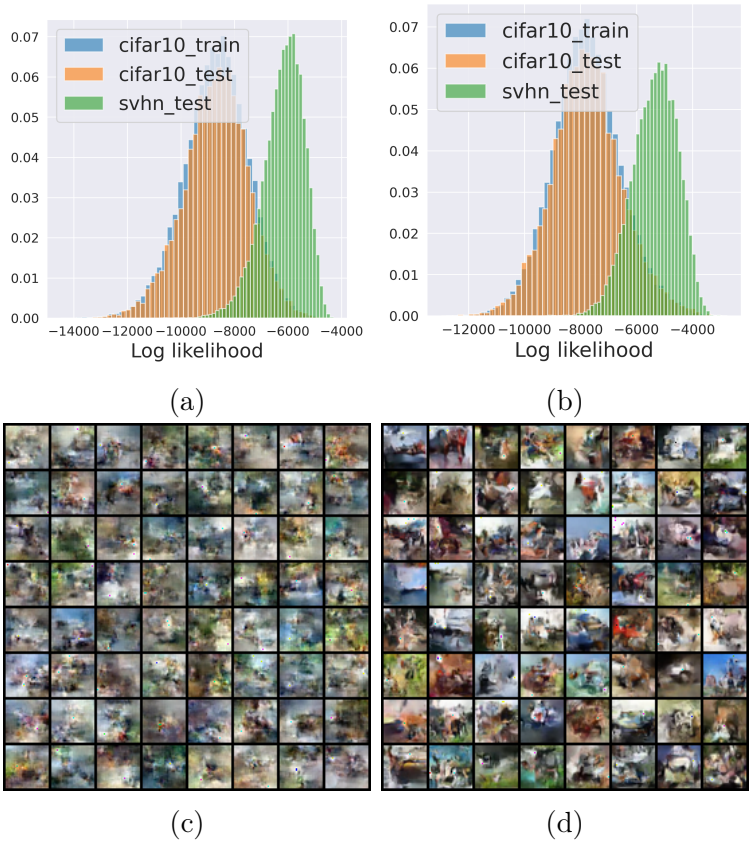


Figure 2.7: Model capacity vs. OOD detection. While the density histograms estimated from the simple Glow (a) and complex Glow (b) are similar, the generation from the simple Glow (c) and complex Glow (d) are noticeably different. More surprisingly, the OOD detection performance by simple Glow (AUROCs: 0.99|0.99) is better than the complex Glow (AUROCs: 0.96|0.95).

run our OOD detection algorithms with SVHN as OOD. Surprisingly, we found that the simpler Glow yields better OOD detection performance. We also observed similar results with RealNVP (see Appendix A.7.5 for details).

Training time: In Appendix A.7.5, we visualize how the generation quality and the OOD detection performance evolve with training time. We use RealNVP with 16 blocks and 512 hidden channels trained on CelebA. Generally, more training time leads to a better model, which is indicated by the generation quality. Taking CIFAR-10 as OOD, we observe that the detection performance improves with training time, as shown in Figure A.4. However, while we have similar observations on most dataset pairs, we still found some anomalies, e.g., Glow on CIFAR-10/SVHN and FMNIST/MNIST, where the detection performance fluctuates or even declines with the training time (see Appendix A.7.5).

We conclude that model accuracy, as measured by the generation quality in our exper-

Table 2.2: Comparison in the sample and latent spaces. AUROCs are shown and segmented by |. S denotes detection in the sample space, while L-R and L-G denote detecting in the latent space using RealNVP and Glow, respectively.

InD/OOD	S | L-R | L-G
FMNIST/MNIST	1.00 | 1.00 | 1.00
CIFAR-10/SVHN	0.90 | 0.97 | 0.99
CIFAR-10/CelebA	0.96 | 0.93 | 0.99
SVHN/CIFAR-10	0.62 | 0.98 | 0.99
SVHN/CIFAR-100	0.67 | 0.98 | 0.99
SVHN/CelebA	0.88 | 0.99 | 1.00

Table 2.3: Comparison in the sample and latent spaces with noise. AUROCs are shown and segmented by |. S denotes detection in the sample space, while L-R and L-G denote detecting in the latent space using RealNVP and Glow, respectively.

InD/OOD	Noise	S | L-R | L-G
CIFAR-10/SVHN	No noise	0.90 | 0.97 | 0.99
	$\mathcal{N}(0, 0.1)$	0.86 | 1.00 | 1.00
	$\mathcal{N}(0, 0.2)$	0.90 | 1.00 | 1.00
	$\mathcal{N}(0, 0.3)$	0.98 | 1.00 | 1.00
	$\mathcal{N}(0, 0.4)$	1.00 | 1.00 | 1.00
SVHN/CIFAR-10	No noise	0.62 | 0.98 | 0.99
	$\mathcal{N}(0, 0.1)$	0.69 | 1.00 | 1.00
	$\mathcal{N}(0, 0.2)$	0.79 | 1.00 | 1.00
	$\mathcal{N}(0, 0.3)$	0.88 | 1.00 | 1.00
	$\mathcal{N}(0, 0.4)$	0.93 | 1.00 | 1.00

iments, is not always positively correlated with the OOD detection performance. These results confirm the findings of Schirrmeister et al. (2020) and L. Zhang et al. (2021) on more models and more real datasets, and reveal the surprising phenomenon that a misestimated model can sometimes lead to better OOD detection. We present our detailed explanation and discussion in Appendix A.7.5.

2.4.5 Distribution Comparison in Latent vs. Sample Spaces

Instead of transforming the data into the latent space, we can directly apply our detection method to the raw inputs without training a flow model, i.e. set $\mathsf{T} = \mathrm{Id}$. Naturally, one wonders if it is necessary to use a flow model for OOD detection. Or, can we detect in the sample space directly? Thanks to the generality of our algorithm, we are positioned to report a fair comparison below.

When performing detection in the sample space directly with our method, we can only use the two-sample version due to the absence of the prior distribution. In this experiment, the batch size is fixed to be 10 and the number of random projections is fixed to be 50. We first run our GOOD2KS (without the inverse transformation) in the sample space on image datasets, and compare with the detection results obtained in the latent space, as shown in Table 2.2. Interestingly, we found that detecting in the latent space is generally better than in the sample space, especially for SVHN/CIFAR-10, SVHN/CIFAR-100, and SVHN/CelebA.

Next, we examine the detection performance by varying the OOD distribution. To this end, we manually inject zero-mean Gaussian noise to the OOD data. The results are

summarized in Table 2.3, where we can see that the detection performance in the sample space varies with the OOD distribution, while that in the latent space is superior and more stable.

We first note that detecting in the original sample space and latent space should be similar (if not the same) without any dimension reduction techniques, since a diffeomorphic transform preserves divergences as we recalled in Section 2.3.2. With random projections, however, we might expect different results. The meaningful/semantic differences in the input space between InD and OOD might be masked by noises or some subtle factors (e.g. background, lighting, and texture in the context of image domains) that will affect the distribution of pixel densities, and these irrelevant variations will be projected into random directions via random projections, making it more challenging to detect the core difference. In contrast, flow models encode semantic and meaningful features of input data into the latent space, where the noise and irrelevant features are compressed or regularized, resulting in more distinct latent distributions of InD and OOD that improve the detection performance. We leave a more comprehensive study of the underlying reason for future work.

2.5 Related Work

OOD detection has been explored from different perspectives, e.g., discriminative methods, generative models, or hypothesis tests; see Chalapathy and Chawla (2019), Pang, Shen, et al. (2021), and Toth and Chawla (2018) for extensive reviews. Below we only discuss papers on unsupervised methods using generative models.

Unsupervised group OOD detection Nalisnick, Matsukawa, Teh, and Lakshminarayanan (2019) propose the typicality test for group OOD detection under the hypothesis that the in-distribution samples are drawn from the typical set of the data distribution, which may not overlap with the regions of high density. This hypothesis is recently interrogated by L. Zhang et al. (2021), who instead attribute the failure of OOD detection in deep generative models to model misestimation. Moreover, J. Song et al. (2019) observe that batch normalization can lower the likelihood of a batch of OOD samples, based on which a permutation test is proposed for group OOD detection. Chalapathy, Toth, et al. (2018) define a group reference function which aggregates the information of input groups and then suggests a distance score for OOD detection that measures the deviation between a test group and the group reference function. It's worth mentioning that both J. Song et al. (2019) and Chalapathy, Toth, et al. (2018) adopt a fixed and large batch size (64 and 1536, respectively) for all experiments, and thus it is not clear how the batch size (especially the smaller ones) would affect the detection performance. Our work is most related to Y. Zhang et al. (2020), who also consider detection in the latent space of flow models. The main differences lie in three aspects: (1) they impose the Gaussian assumption on the latent distribution of test samples, so that they can compare it with the latent prior in a closed form using KL-divergence. In contrast, our KS test is non-parametric, hence does

not require any distributional assumption on test data; (2) they assume the trained flow model is well-trained, while we additionally provide GOOD2KS for under-trained models; and (3) we use random projections to consider both marginal distribution and correlations of all dimensions, while they need to explicitly estimate the correlation coefficients to quantify the inter-dimensional correlations, which can be challenging in practice.

Unsupervised point OOD detection Compared to group OOD detection, point OOD detection treats test samples individually. In the context of deep generative models, many researchers have tried to explain the failure of log-likelihood for OOD detection from different perspectives, e.g., the background statistics (Ren et al. 2019), the inductive biases (Kirichenko et al. 2020), the input complexity (Serrà et al. 2020), and the model accuracy (Choi et al. 2018), and proposed new metrics therein. Our work can also be extended to point detection by artificially creating a batch of samples for each individual test sample (see Appendix A.6 for more details). Note that Havtorn et al. (2021) and Schirrmeister et al. (2020) also develop detection scores based on (partial) latent variables of flow model and VAE, respectively. Differently, Havtorn et al. (2021) interpret their likelihood ratio test as comparing divergence between true and approximate posterior in latent space (since true posterior is intractable), whereas Schirrmeister et al. (2020) contrast the log likelihood of the last-scale latent code that encodes global object-specific features with that of some background latent code. Nevertheless, both approaches do not employ random projections or KS tests as we do. Morningstar et al. (2021) extended the typicality test to the pointwise setting, by fitting a kernel density estimator to each of a few test statistics on the training set and then computing the sum of log probability of each statistic as the detection score for a given data point.

There are a few recent works leveraging diffusion models in OOD detection tasks. The idea is to corrupt the input image (e.g. by random mask (Liu, J. P. Zhou, et al. 2023) or random noise injection (Graham et al. 2023)) and then reconstruct the input by a backward diffusion process, where the InD images are expected to have smaller reconstruction errors compared to OOD images.

Representation learning for OOD detection OOD detection is challenging in high dimensions. This motivates the study of representation learning, which aims to characterize discriminative information of the input with a reduced dimension for the OOD detection. Example methods include projection (Pang, L. Cao, et al. 2018; Pevnỳ 2016), feature selection (Azmandian et al. 2012; Pang, L. Cao, et al. 2017), and deep learning based methods. For the latter one, representations can be obtained either from a pre-trained model (Andrews et al. 2016; Pang, Yan, et al. 2020; Tudor Ionescu et al. 2017; J. T. Zhou et al. 2019), or be directly learned with a neural network such as autoencoders (Erfani et al. 2016; Ionescu et al. 2019; J. Wang et al. 2019; Xu, Ricci, et al. 2015). In our work, the representation is obtained by random projecting the latent space of flow models. Compared with other deep learning based approaches, our method requires no additional networks and is computationally more efficient.

2.6 Summary

In this work, we re-examined the potential of flow models for OOD detection. We provided practical OOD detection algorithms that compare distributional information in the latent space and impose no parametric assumption. We compared with SOTA benchmarks and obtained comparable and more stable performances. Experimentally, we demonstrated the benefits of detecting OOD data in the latent space and confirmed that OOD detection performance is not always positively correlated with model accuracy. In the future, we would like to characterize OOD distributions that can be reliably detected by our methods, as well as the possibility of projecting onto high dimensional subspaces.

Chapter 3

DP-LFlow: Differentially Private Latent Flow for Scalable Sensitive Image Generation

3.1 Preface

While DP-SGD is the dominating algorithm in DP deep learning, it fails to scale well on larger supervised models given the same privacy budgets, partly because of the high model/gradient dimension as investigated in D. Yu et al. (2021). The issue also exists for generative models, thus restricting its model utility on large input (which generally requires more complex models) and stronger DP guarantees (more noise). In this work, to avoid excess gradient perturbation in DP-SGD, we propose to train a generative model in a lower dimensional latent space. The proposed method is shown to be scalable to stronger DP guarantees and high-resolution image datasets.

Abstract: Privacy concerns grow with the success of modern deep learning models, especially when the training set contains sensitive data. Differentially private generative model (DPGM) can serve as a solution to circumvent such concerns by generating data that are distributionally similar to the original data yet with differential privacy (DP) guarantees. While GAN has attracted major attention, existing DPGMs based on flow generative models are limited and only developed on low-dimensional tabular datasets. The capability of *exact* density estimation makes the flow model exceptional when density estimation is of interest. In this work, we will first show that it is challenging (or even infeasible) to train a DP-flow via DP-SGD, i.e. the workhorse algorithm for private deep learning, on high-dimensional image sets with acceptable utility, and then we give an effective solution by reducing the generation from the pixel space to a lower dimensional latent space. We show the effectiveness and scalability of the proposed method via extensive experiments, where the

proposed method achieves a significantly better privacy-utility trade-off compared to existing alternatives. Notably, our method is the first DPGM to scale to high-resolution image sets (up to 256×256). Our code is available at `https://github.com/dihjiang/DP-LFlow`.

The following sections are the main paper and appendix of D. Jiang and S. Sun (2023). "DP-LFlow: Differentially private latent flow for scalable sensitive image generation". *Transactions on Machine Learning Research*.

3.2 Introduction

Large-scale datasets (Bennett, Lanning, et al. 2007; Deng et al. 2009; Lewis et al. 2004) facilitate the great success of modern machine learning (ML) systems. However, privacy concerns arise especially when sensitive data (e.g. human face images and medical data) are involved in the training. Prior privacy-preserving techniques include naive data anonymization (Narayanan and Shmatikov 2008), k-anonymity (Sweeney 2002), l-diversity (Machanavajjhala et al. 2007), semantic security (Goldwasser and Micali 1984), and differential privacy (DP) (Dwork 2006), where DP is recognized as a rigorous quantization of privacy, and it has become the gold-standard privacy-preserving technique in the current ML community.

Differentially private generative model (DPGM) aims to synthesize data that are distributionally similar to the private data while satisfying DP guarantees so that the individual privacy that can be inferred from the incremental change in the dataset is bounded. DPGM can (1) serve as a proxy for releasing private data, and can (2) generate data for private data analysis tasks (e.g. data querying and ML tasks) without incurring further privacy cost, as ensured by the post-processing theorem (Dwork, Roth, et al. 2014).

Generative adversarial network (GAN) (Goodfellow et al. 2014) has attracted the most attention in developing DPGMs (Augenstein et al. 2020; D. Chen et al. 2020; Jordon et al. 2019; Long et al. 2021; Torkzadehmahani et al. 2019; Xie et al. 2018), and VAE also draws a lot of attention (Acs et al. 2018; Bernau et al. 2022; Q. Chen et al. 2018; Pfitzner and Arnrich 2022; Takahashi et al. 2020; Weggenmann et al. 2022). In contrast, the DPGMs based on the normalizing flow are relatively limited (Lee et al. 2022; Waites and Cummings 2021). Due to the capability of the *exact* density computation, flow models are particularly useful when density is of interest in some applications. For example, developing a DP-flow could benefit performing anomaly detection in a privacy-preserving manner (Waites and Cummings 2021). However, both existing DP-flow works (Lee et al. 2022; Waites and Cummings 2021) are restricted to tabular datasets (with lower data dimensions). It remains unclear whether DP-flow is feasible on higher-dimensional datasets such as image datasets.

Abadi et al. (2016) propose Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm, which has become the standard approach to train a DP deep learner. The core steps of DP-SGD are clipping the per-example gradient norm and injecting

Calibrated Gaussian noise into the aggregated gradient of a batch. Briefly, let p denote the model dimension, the noise introduced by DP-SGD in each iteration is given by $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_p, \sigma^2 C^2 \mathbb{I}_{p\times p})$, where C is the clipping bound of gradient norm and σ is a noise multiplier. Apparently, $\mathbb{E}[\|\mathbf{z}\|_2^2] = p\sigma^2 C^2 \propto p$, which means the model utility may not be preserved with DP-SGD for a large model. For example, D. Yu et al. (2021) show that the gradient will be submerged in the added noise (by DP-SGD) when the model becomes larger on a series of ResNet (He et al. 2016) variants. This utility drop could become more pronounced under strong privacy guarantees (e.g. $\epsilon = 1$), as the noise multiplier becomes larger. Note that according to ϵ-DP definition (given in Section 3.3), the privacy protection becomes weak when $\epsilon = 10$ or above, because the ratio of two probabilities is upper bounded by $\exp(10) \approx 2.2 \times 10^4$, whereas they are presumed to be comparable for practical deployment (e.g. $\epsilon = 1$ or below). However, a line of recent SoTA DPGMs only generate acceptable images at $\epsilon = 10$, whereas they are not able to scale well on small ϵ (e.g. $\epsilon = 1$) (D. Chen et al. 2020; Jordon et al. 2019; Torkzadehmahani et al. 2019).

Training a normalizing flow with DP-SGD on an image set seems ostensibly easy, but any ML researcher/engineer will encounter the two non-trivial empirical challenges, as also pointed out by Lee et al. (2022): (1) *batch normalization (BN) challenge*: flow models usually apply a batch normalization (BN) layer in each block to boost the performance, where the per-example gradient (in DP-SGD) is not available, as BN will break the independence among all gradients in a batch; (2) *model complexity challenge*: flow models generally consist of repetitive blocks of invertible transformations with a large depth, which results in higher model complexity compared to other generative models. We explore the challenges with two SoTA flow models, i.e. RealNVP (Dinh et al. 2017) and Glow (Kingma and Dhariwal 2018). Glow uses activation normalization as an alternative to BN, thus per-example gradient is computable (i.e. no BN challenge for Glow). However, both RealNVP and Glow suffer from the model complexity challenge, i.e. they tend to be more complex than other generative model counterparts for synthesizing images with similar fidelity, thus are more challenging to attain desirable model utility with DP-SGD. Our motivating example is shown in Figure 3.6, where naively training a flow generative model in the pixel space with DP-SGD results in null model utility.

Our contributions can be summarized as follows:

- We explore the practical challenges of training a DP flow via DP-SGD on image sets, and propose an efficient and effective solution, i.e. differentially private latent flow (DP-LFlow), by reducing the training of flow from the full pixel space to a lower-dimensional latent space, which is more resilient to the noise perturbation (by DP-SGD).
- Training DPGM on high-resolution images (256×256 pixels and beyond) is extraordinarily challenging due to the inevitably increased model dimension, and to our best knowledge, none of the existing related works attempt to do so. In this work, we will show that DP-LFlow is also scalable to the high-resolution image (256×256) generation with DP constraints.

- The proposed method yields state-of-the-art (SoTA) performance on model utility under the same (ϵ, δ)-DP constraint on widely compared image benchmarks. Moreover, our method indicates more robustness and scalability on different datasets (gray-scale and RGB) and different DP constraints.

3.3 Preliminary

In this section, we recall background knowledge in differential privacy.

3.3.1 Differential Privacy

Differential privacy is widely regarded as a rigorous quantization of privacy, which upper-bounds the deviation in the output distribution of a randomized algorithm given an incremental deviation in the input. Please refer to Definition 1 for its formal definition.

There is a convenient parallel composition theorem for ϵ-DP mechanisms:

Theorem 4 (Parallel composition theorem of ϵ-DP, (McSherry 2009))**.** *Let $\mathcal{M}_i$ $(i = 1, 2, \ldots, k)$ be k DP mechanisms, and each $\mathcal{M}_i$ satisfies ϵ_i-DP. Given a deterministic partitioning function f, let $D_1, D_2, \ldots, D_k$ be the disjoint partitions by executing f on D. Releasing $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_k(D_k)$ satisfies $\max_{i \in \{1,2,\ldots,k\}} \epsilon_i$-DP.*

We will extend the above parallel composition to the (ϵ, δ)-DP notion in Section 3.4.

A famous theorem, i.e. post-processing theorem, which is utilized by existing works (as well as ours) for proving DP guarantee of a published model, is given by Theorem 1. Sampling from a DPGM is independent of training data, thus can be viewed as a post-processing step and does not breach the DP guarantee.

Rényi differential privacy (RDP) extends ordinary DP using Rényi's α divergence (Rényi 1961) and provides tighter and easier composition property than the ordinary DP notion, thus we adopt RDP to accumulate the privacy cost. We defer details of RDP and implementation to Appendix B.5.

3.3.2 DP-SGD

Within predetermined training iterations, in each iteration, DP-SGD (Abadi et al. 2016) subsamples a batch from the training set, clips and perturbs the gradient as in Algorithm 2, and optimizes the model with privatized gradient $\tilde{g}_\theta$. As mentioned earlier, the norm of the Gaussian noise introduced at line 4 in Algorithm 2 will scale linearly with the model dimension, thus will generally degrade the utility of large models.

Algorithm 2: Gradient perturbation in DP-SGD

Input: Private training set $X = \{\mathbf{x}_i\}_{i=1}^N$, loss function $\mathcal{L}(\cdot)$, batch size B, noise multiplier σ, gradient clipping bound C, model parameter θ

for $i \leftarrow 1$ ***to*** B **do**

  $g_\theta(\mathbf{x}_i) = \nabla_\theta \mathcal{L}(\mathbf{x}_i; \theta)$

  $g_\theta(\mathbf{x}_i) = g_\theta(\mathbf{x}_i) \cdot \min\left(1, \frac{C}{\|g_\theta(\mathbf{x}_i)\|_2}\right)$

$\tilde{g}_\theta = \frac{1}{B}\left[\sum_{i=1}^B g_\theta(\mathbf{x}_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbb{I})\right]$

3.3.3 Flow-based Generative Models

We briefly recap the flow generative models. Flow models learn a bijective map T between a simple prior distribution q_0 (e.g. Gaussian) and the target distribution q: $\mathbf{z} \sim q_0 \Leftrightarrow \mathsf{T}(\mathbf{z}) \sim q$. Through the change-of-variable formula, the log-likelihood of input is tractable:

$$q_\mathsf{T}(\mathbf{x}) = q_0(\mathbf{z}) \left| \det \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\mathbf{x}} \right| = q_0\big(\mathsf{T}^{-1}(\mathbf{x})\big) \left| \det \frac{\mathrm{d}\mathsf{T}^{-1}(\mathbf{x})}{\mathrm{d}\mathbf{x}} \right|$$

Parameterize the bijective map by neural networks, we can train flow models by minimizing the Kullback–Leibler (KL) divergence between the true and estimated distribution:

$$\begin{aligned} \min_\mathsf{T} \mathbb{D}_{KL}(q(\mathbf{x}) || q_\mathsf{T}(\mathbf{x})) &= \min_\mathsf{T} \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{q_\mathsf{T}(\mathbf{x})} \, \mathrm{d}\mathbf{x} \\ &= \min_\mathsf{T} \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})}[-\log q_\mathsf{T}(\mathbf{x})] - \mathbb{H}[q] \end{aligned}$$

where $\mathbb{H}[q]$ is the entropy of true distribution. Therefore, training the flow model amounts to minimizing the negative log-likelihood of input.

3.4 Method: DP-LFlow

As shown in D. Yu et al. (2021), a smaller/simpler ResNet is more resilient to DP-SGD. The intuition is that the model utility will saturate as the model complexity increases. Therefore, an over-complicated model may (unnecessarily) be disturbed by more noises in DP-SGD. We first confirm this insight for DPGMs as well. Consider privately training a VAE on MNIST (LeCun et al. 1998) for example. Figure 3.1 shows such comparison under non-private and private settings (with different privacy costs), respectively, where the generation quality is measured by Fréchet Inception Distance (FID) (Heusel et al. 2017) (lower is better). Figure 3.1 indicates that a smaller VAE generates better images than larger counterparts (both qualitatively and quantitatively) under the DP training, even though larger VAEs perform better in the non-DP setting. In fact, shrinking the model size under the DP training will benefit from the following aspects: (1) smaller models are

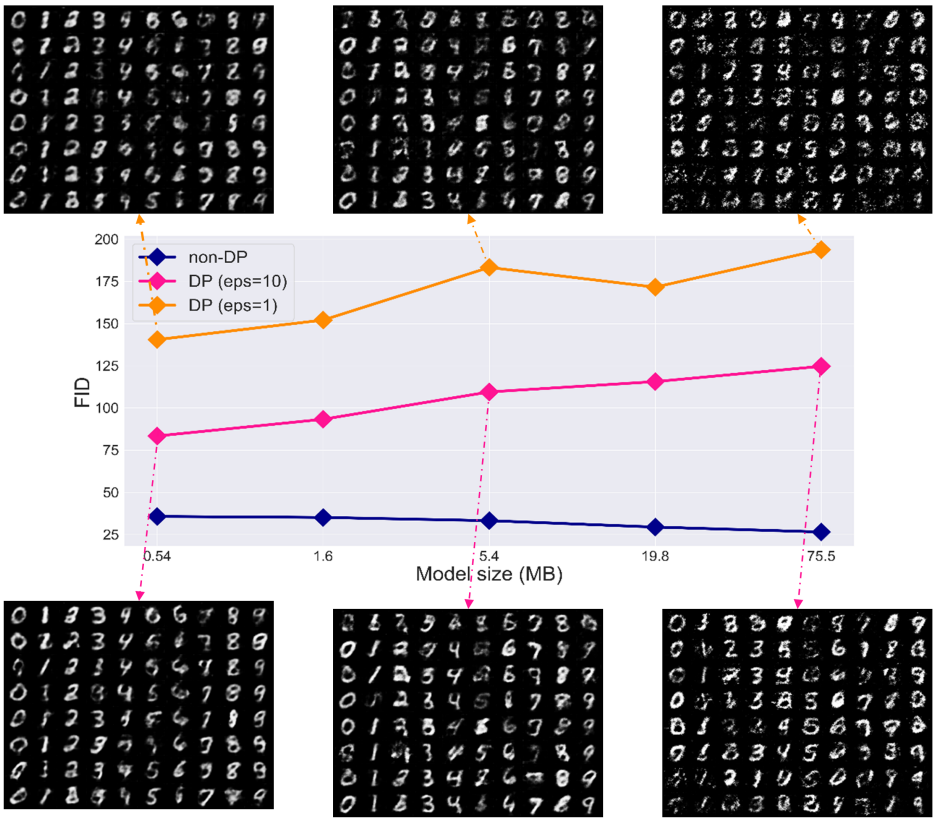


Figure 3.1: FID vs. VAE size (in MB). We only vary the model complexity, with all the rest training parameters (e.g. subsampling rate, noise multiplier, training iterations) fixed.

more resilient to (larger) noises (associated with strong DP guarantees); (2) a significant training time overhead remains a notable challenge for DP-SGD (Subramani et al. 2021) due to the gradient clipping and randomization. Smaller models could facilitate more efficient DP training; (3) as suggested by prior work (De et al. 2022), a larger batch size in DP-SGD contributes to better model utility. With limited GPU memory, smaller models allow for a large training batch size, since both input and model are sent to GPU for training in practice.

Latent Flow: However, the model expressiveness will be restricted if the model is too simple. Therefore, we aim to design a model that is small yet expressive enough so that we can achieve a better privacy-utility trade-off with DP-SGD. Inspired by the recent latent diffusion model (Rombach et al. 2022) that achieves SoTA text-to-image generation performance via reducing the diffusion process from the raw input space to a lower dimensional latent space, we propose to train a normalizing flow in a similar manner (i.e. on the latent code produced by an autoencoder), by simultaneously minimizing the reconstruction loss of the autoencoder and the negative log-likelihood of the flow. Specifically, let ϕ, θ, ω

parameterize the encoder, decoder, and flow, respectively, and we adopt the notations in Section 3.3.3. Let $\mathbf{w} = f_\phi(\mathbf{x})$ be the latent embedding of the autoencoder, the training objective can be written as:

$$\min_{\phi,\theta,\omega} \mathbb{E}_{\mathbf{x}\sim q}[\underbrace{\|\mathbf{x} - D_\theta(\mathbf{w})\|_2^2}_{\text{Autoencoder loss}} \times T^2 \underbrace{-\big(\log q_0(\mathbf{z}) + \log \Big| \det \frac{\mathrm{d}\mathsf{T}_\omega^{-1}(\mathbf{w})}{\mathrm{d}\mathbf{w}}\Big|\big)}_{\text{Normalizing flow loss}}]$$

where $\mathbf{z} \sim q_0 = \mathcal{N}(0, \mathbb{I})$ and T is a temperature parameter. As shown in Rombach et al. (2022), the semantic meaning of most images still remains after aggressive compression, thus allowing us to train a flow in an aggressively trimmed latent space, which avoids unnecessary and expensive computation on full input dimensions. It is worth mentioning that latent flow is also not sensitive to BN layers (e.g. for RealNVP), i.e. the utility of latent flow is slightly reduced by removing the BN layer, which validates the use of DP-SGD for latent flow models. Under the non-private setting, the proposed model with a configuration of size 2.5 MB achieves an FID = 12.5 on MNIST, which is already superior to VAE counterparts with a larger model size (e.g. in Figure 3.1), suggesting its promising usage under the private training.

Partitioning Dataset: Current SOTA methods tend to apply conditional generative models (conditioned on labels), where the label is encoded in the model as part of the input, thus the noise perturbation also distorts the label information, which is unnecessary. We run DP-SGD on the proposed model under the conditional setting (autoencoder + conditional flow), and observe that the label information is largely distorted when $\epsilon = 1$ (see Section 3.5.5 for details). To circumvent the perturbation to labels, we propose to *partition the dataset according to labels*, train unconditional generative models on each of the subsets, and release the union of all unconditional generators as the resulting model. The partitioning is also beneficial for shrinking the model size, as each generator is only interactive with a sole data modality instead of multi-modalities. Adapted from proposition 2.5 in (N. Li, Lyu, et al. 2016), the DP guarantee for the union can be derived from the parallel composition, by extending the original parallel composition theorem (Theorem 4) from ϵ-DP notion to (ϵ, δ)-DP notion. The proof can be found in Appendix B.1.

Theorem 5. *Let $\mathcal{M}_i$ $(i = 1, 2, \ldots, k)$ be k DP mechanisms, and each $\mathcal{M}_i$ satisfies (ϵ_i, δ_i)-DP. Given a deterministic partitioning function f, let $D_1, D_2, \ldots, D_k$ be the disjoint partitions by executing f on D. Releasing $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_k(D_k)$ satisfies $(\max_{i\in I} \epsilon_i, \max_{i\in I} \delta_i)$-DP, where $I = \{1, 2, \ldots, k\}$.*

For simplicity, we set ϵ_i, δ_i the same as the target ϵ, δ for all sub-models in the experiment. The schematic workflow of DP-LFlow is shown in Figure B.1.

3.5 Experiments

In this section, we evaluate and compare DP-LFlow against SoTA baselines through extensive experiments in Section 3.5.2. More importantly, we will show that DP-LFlow is amenable to high-resolution image sets in Section 3.5.3, which was hardly studied in prior related works. Besides, since flow models are capable of computing the exact likelihood of input, they are sometimes more applicable when density estimation is of concern. Section 3.5.4 shows such an example, where DP-LFlow can detect intra-dataset anomaly with DP guarantees. RealNVP is used as the flow model in DP-LFlow, as it yields better performance in practice. Implementation details and neural network configurations are given in Appendix B.5.

3.5.1 Experimental Setup

Datasets: We consider three widely used image datasets, including both grayscale images (MNIST (LeCun et al. 1998), Fashion MNIST (Xiao et al. 2017)) and RGB images (CelebA (Liu, P. Luo, et al. 2015)), as well as one high-resolution RGB datasets (CelebA-HQ (Karras et al. 2018), for our presentation only). For MNIST and Fashion MNIST, we generate images conditioned on 10 respective labels. For CelebA and CelebA-HQ, we condition on gender. Descriptions and preprocessing of the datasets are given in the Appendix B.2.

Evaluation Tasks & Metrics: We evaluate and compare DPGMs by two metrics via 60k generated images:

- Fréchet Inception Distance (FID) (Heusel et al. 2017).
- Classification accuracy. We train three different classifiers, e.g. logistic regression (LR), multi-layer perceptron (MLP), and convolutional neural network (CNN), on generated images, then test the classifier on real images. We take 5 runs and report the average. See Appendix B.5 for details of classifiers.

Post-processing: We observe that FID is quite sensitive to the noise around the object (e.g. digits, cloths) and the sharpness of images, even when the semantic content is retained. We apply a post-processing step to our DP image generation by first smoothing and then sharpening the generated images, which could lead to $\sim$10 improvement in FID. Whereas this post-processing does not improve classification accuracy too much. Adjusting the blurriness and sharpness of generation is not involved with training data, thus is considered a post-processing step that does not breach the DP guarantee by Theorem 1. We will report our quantitative evaluations with and without this trick.



Figure 3.2: Qualitative comparison on MNIST and Fashion MNIST under $(10, 10^{-5})$-DP. Images of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from (T. Cao et al. 2021). Images of G-PATE and DataLens are cited from their papers, respectively.

SoTA Baselines: Our method is compared with the following baseline methods that are also developed on image datasets, i.e. DP-CGAN (Torkzadehmahani et al. 2019), DP-MERF (Harder et al. 2021), Datalens (B. Wang, Wu, et al. 2021), PATE-GAN (Jordon et al. 2019), G-PATE (Long et al. 2021), GS-WGAN (D. Chen et al. 2020), DP-Sinkhorn (T. Cao et al. 2021). For more details, we refer readers to Section 3.7 and respective references.

3.5.2 Comparison with SoTA Baselines

We follow the general benchmark for evaluating DP generation, by considering both weak (e.g. $\epsilon = 10$) and strong (e.g. $\epsilon = 1$) privacy guarantees. The proposed DP-LFlow is compared with SoTA baselines through extensive qualitative and quantitative experiments on both grayscale and RGB image datasets. For completeness, we also report our quantitative results without the post-processing trick, where the relative rank of our method without the post-processing trick remains the same as with the trick.

3.5.2.1 Under the Weak DP Guarantee $\epsilon = 10$

Most existing works perform reasonably when $\epsilon = 10$, as shown in Figure 3.2. Nevertheless, as shown in Table 3.1, DP-LFlow achieves significant improvement in both FID and classification accuracy.

Table 3.1: Quantitative comparison on MNIST and Fashion MNIST given $(10, 10^{-5})$-DP. Acc denotes classification accuracy, which is shown in %. ↑ and ↓ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021). Results of G-PATE and DataLens are cited from their papers, respectively. Ours* denotes our result without the post-processing trick.

Method	MNIST				Fashion MNIST			
	FID ↓	LR Acc ↑	MLP Acc ↑	CNN Acc ↑	FID ↓	LR Acc ↑	MLP Acc ↑	CNN Acc ↑
Real data (non-private)	1.6	92.2	97.5	99.3	2.5	84.5	88.2	90.8
DP-CGAN	179.2	60	60	63	243.8	51	50	46
DP-MERF	121.4	79.1	81.1	82.0	110.4	72.3	70.8	73.2
G-PATE	150.6	-	-	80.9	171.9	-	-	69.3
DataLens	173.5	-	-	80.7	167.7	-	-	70.6
GS-WGAN	61.3	79	79	80	131.3	68	65	65
DP-Sinkhorn	55.6	79.1	79.2	79.1	129.4	70.2	70.2	68.9
Ours	**16.8**	**85.8**	**93.0**	**95.3**	**72.3**	**78.6**	**78.8**	**81.0**
Ours*	25.4	85.1	92.4	94.8	80.8	78.1	78.4	80.7

3.5.2.2 Under the Strong DP Guarantee $\epsilon = 1$

When $\epsilon = 1$, i.e. the more challenging case under the DP setting, we consider both gray-scale image datasets and RGB image datasets for comparison. Figure 3.3 and Figure 3.4 show that the existing works are not readily amenable to a small ϵ such as 1. In contrast, DP-LFlow exhibits significant visual improvement on all three datasets.

Quantitatively, Table 3.2 indicates that DP-LFlow outperforms other baselines on CNN classification accuracy on all three datasets. Besides, DP-LFlow achieves FID scores that are comparable to SoTA performance on Fashion MNIST and achieves the best FID on MNIST and CelebA. Given that the three datasets are either largely different in input dimension or largely different in semantic content, the superior performance of DP-LFlow further validates its generality and versatility.

It is worth mentioning that a couple of recent works on DP diffusion models are also capable of generating realistic images under DP constraints and indicate superiority on downstream ML tasks (Dockhorn et al. 2023; Ghalebikesabi et al. 2023). For example, when $\epsilon = 1$, Dockhorn et al. (2023) achieve 93.4 and 73.6 test classification accuracies on MNIST and FMNIST, respectively, while ours are 88.9 and 77.4. However, their required computational resource is significantly higher than DP-LFlow, e.g. Dockhorn et al. (2023) need 8 GPUs and one day to train a DP diffusion model on MNIST and FMNIST, while our method only requires 1 single GPU and a few hours; On CelebA, Dockhorn et al. (2023)

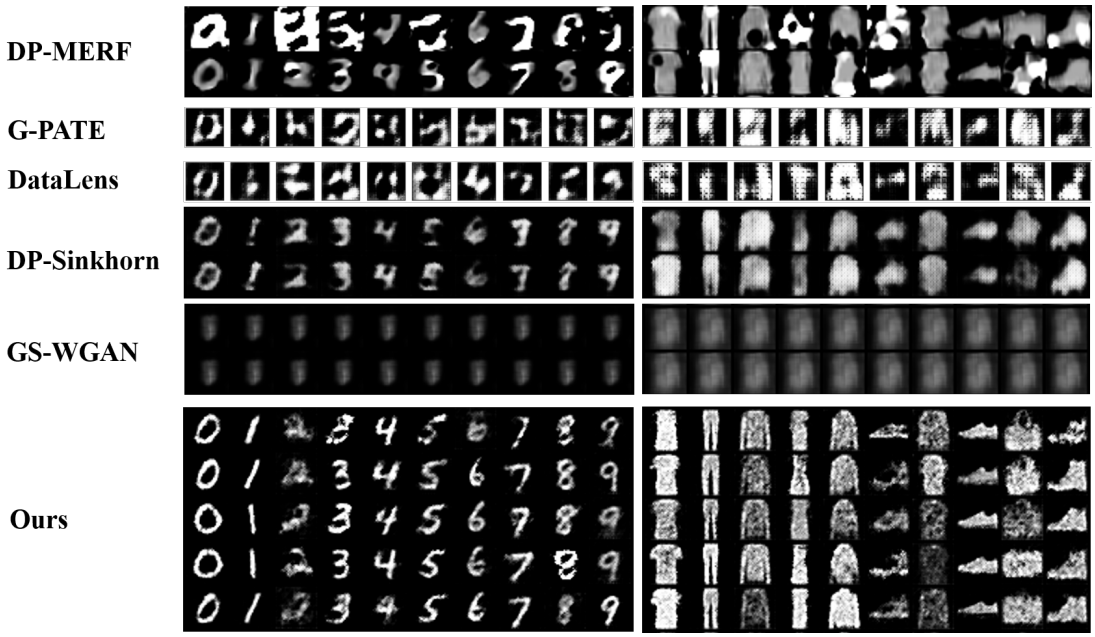


Figure 3.3: Qualitative comparison on MNIST and Fashion MNIST under $(1, 10^{-5})$-DP. Images of G-PATE and DataLens are cited from their papers, respectively.

needs 8 GPUs and 4 days, while our method only needs 1 GPU and around half-day. More importantly, DP-LFlow still has two unique properties compared to DP diffusion models: (1) DP-LFlow can generate large images up to $3 \times 256 \times 256$ (see Section 3.5.3), while Dockhorn et al. (2023) and Ghalebikesabi et al. (2023) only present generation results on images up to $3 \times 32 \times 32$; (2) DP-LFlow is able to conduct the DP-OOD detection task (see Section 3.5.4).

3.5.3 Generating High-resolution Images with DP Constraints

To our best knowledge, the highest resolution image dataset used in related works is CelebA downsampled at 64×64 pixels (by DataLens and G-PATE), which, however, is still not typically considered a high-resolution space. Here we consider a high-resolution dataset CelebA-HQ in 256×256, which is also close to many modern large-scale image classification benchmark datasets such as ImageNet (224×224 or 256×256).

Note that now the input dimension drastically increases from $1 \times 28 \times 28 = 784$ (for MNIST and Fashion MNIST) to $3 \times 256 \times 256 \approx 1.97 \times 10^5$. This means that the model generally has to drastically scale up to adequately learn the input distribution, leading to a significant challenge for training a DPGM with DP-SGD.

Nevertheless, with the help of lower dimensional latent space where we can apply aggressive compression while retaining the semantic content, we can restrict the generative model to a size ($\sim$ 22 MB, compared to $\sim$ 2.5 MB on MNIST) that is suitable for DP train-



Figure 3.4: Qualitative comparison on CelebA conditioned on gender under $(1, 10^{-5})$-DP. Images of DataLens are cited from the original paper. Top row: female. Bottom row: male.

ing yet adequate to learn the data distribution. By training a DP-LFlow on CelebA-HQ, Figure 3.5 shows that DP-LFlow is able to produce diverse and recognizable face images with DP constraints on such high dimensional input space.

3.5.4 Differentially Private Out-of-distribution (OOD) Detection

One exceptional feature of flow models is their capability of *exact* density estimation, which can benefit certain applications such as out-of-distribution detection. A natural method to do so is to thresholding the log-likelihood, since it is presumed that the in-distribution (InD) data (where the training data are drawn from) will be assigned a higher likelihood than OOD data by the flow model. This idea can be readily extended to latent flow models by checking the log-likelihood of latent code mapped from the input data. As our sub-models are privately trained on each subset by class, we can immediately conduct the differentially private intra-dataset OOD detection, by treating each training class as the InD and the rest classes combined as OOD. A lower log-likelihood of the latent code computed from input data implies OOD-ness. We use Area Under Receiver Operating Curve (AUROC) as the evaluation metric, which is threshold-free and widely used in OOD detection literature. Table 3.3 indicates that under weak and strong guarantees, DP-LFlow can effectively detect intra-dataset OOD input on both MNIST and Fashion MNIST across all InD classes in a privacy-preserving manner.

Table 3.2: Quantitative comparison on image datasets given $(1, 10^{-5})$-DP. FMNIST denotes Fashion MNIST. Acc denotes CNN classification accuracy, which is shown in %. ↑ and ↓ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of PATE-GAN and G-PATE are cited from Long et al. (2021). Results of DataLens are cited from B. Wang, Wu, et al. (2021). Ours* denotes our result without the post-processing trick.

Method	MNIST		FMNIST		CelebA	
	FID ↓	Acc ↑	FID ↓	Acc ↑	FID ↓	Acc ↑
PATE-GAN	231.7	41.7	253.2	42.2	434.5	44.5
DP-MERF	118.3	80.5	104.2	73.1	219.4	57.6
G-PATE	154.3	58.8	214.8	58.1	293.2	70.2
DataLens	186.1	71.2	195.0	64.8	297.7	70.6
DP-Sinkhorn	168.5	60.5	184.2	62.1	-	-
Ours	**69.7**	**88.9**	132.4	**77.4**	**204.8**	**73.2**
Ours*	83.4	88.2	143.5	76.8	217.7	72.1

3.5.5 Ablation Study

DP-LFlow contains two main components, and we conduct an ablation study as follows.

3.5.5.1 Flow in the Pixel Space vs. Latent Space

We first trained a RealNVP with DP-SGD by removing the BN layer. Figure 3.6 show that the generation from a DP-RealNVP (subfigure (c)) is largely submerged in the noise. Besides, we also trained a Glow with DP-SGD (with various model complexities), yet ended up with null synthesis when targeting $(10, 10^{-5})$-DP. Both RealNVP and Glow suffer from the model complexity challenge. Compared to the generation of DP-LFlow in Figure 3.4 (which even targets a stronger privacy regime, i.e. $\epsilon = 1$), training a flow in the latent space relieves the model complexity challenge, thus is more resilient to the noise perturbation introduced by DP-SGD.

3.5.5.2 With vs. Without Partitioning the Dataset

Qualitatively, we observe that training a conditional DP-LFlow on the whole dataset will fail to correctly condition on each label, (especially under $\epsilon = 1$), i.e. the generations conditioned on different labels are not consistent with the true labels, thus the numerical classification accuracy is relatively lower, although FID is fairly comparable. We summarize the numerical comparison in Table 3.4.



Figure 3.5: Samples from DP-LFlow trained on CelebA-HQ under $(10, 10^{-5})$-DP. Top 2 rows: female. Bottom 2 rows: male. FID = 328.6, LR classification accuracy = 77.4.

The reason is that, in contrast to uninvertible models like conditional GAN and conditional VAE that only encode labels in the latent vector, the label information in the conditional flow is encoded in all coupling layers, because flow is an invertible model so that both forward and backward pass needs label information. Thus, training conditional flow with DP-SGD will distort the label information more than uninvertible models. We also tried to modify the way of encoding labels in different variants: only encoding labels in the first, last, or both first and last coupling layers, or only encoding labels in the latent vector, but none of the variants solve the issue. Instead, resorting to partitioning the dataset can readily bypass the label distortion in the private training, which leads to better model utility compared to a single conditional model on all classes, and by Theorem 3.1 we can retain the same level of privacy guarantees as publishing a single conditional generator. In other words, partitioning the dataset contributes to improving the privacy-utility trade-off.

We note there is a potential limitation, since each autoencoder learns data representations from separate classes, which may restrict the generalization capability of each individual autoencoder. A straightforward remedy is to train one single autoencoder for all classes, which, however, makes the parallel composition invalid because the single autoencoder will depend on all classes (e.g. changing one point in a class will affect the single autoencoder that is trained for all classes).

3.6 Discussion

Pretraining the autoencoder: Though our current training proceeds by simultaneously updating the autoencoder and flow, one may wonder whether we can pretrain an

Table 3.3: Differentially private OOD detection results. The evaluation metric is shown by AUROC (higher is better, and 1.0 means InD and OOD likelihood are perfectly separable).

InD class	MNIST		Fashion MNIST	
	$\epsilon = 10$	$\epsilon = 1$	$\epsilon = 10$	$\epsilon = 1$
0	0.98	0.87	0.90	0.87
1	0.99	0.98	0.97	0.96
2	0.86	0.75	0.90	0.88
3	0.88	0.82	0.91	0.91
4	0.95	0.85	0.87	0.85
5	0.90	0.75	0.91	0.86
6	0.97	0.77	0.84	0.82
7	0.93	0.88	0.97	0.97
8	0.89	0.74	0.86	0.79
9	0.95	0.83	0.94	0.93
Average	0.93	0.82	0.91	0.88

Table 3.4: Ablation on partitioning the MNIST dataset. We compare the performance without post-processing tricks (smoothing and sharpening) for simplicity.

	ϵ	FID ↓	LR Acc ↑	MLP Acc ↑	CNN Acc ↑
With partitioning	10	25.4	85.2	**92.7**	**95.0**
Without partitioning	10	**21.3**	**86.2**	91.5	91.8
With partitioning	1	**83.4**	**83.2**	**82.4**	**88.3**
Without partitioning	1	87.1	40.8	72.5	77.5

autoencoder first and then train a flow with the pretrained autoencoder. However, there are a few concerns that impede the two-stage training paradigm:

- **Pretraining on a public dataset:** The majority of existing works (including all of our compared works) do not use external public dataset for pretraining, so it would make the comparison unfair if we do so. Besides, we need to put a strong assumption that the public dataset does not have any privacy concerns. While this can be easily ensured in experiments, it may not always be true in reality, thus is likely to undermine the potential impact of this work. Therefore, we stick to restricting our information to the private dataset only.
- **Pretraining on the private dataset:** In this case, changing a point in the dataset will result in a different pretrained autoencoder, which means all the gradients in a batch (when running DP-SGD on flow) will change, instead of at most one gradient



Figure 3.6: Training a RealNVP (∼150 MegaBytes (MB)) on CelebA (a) non-privately with BN layers, (b) non-privately without BN layers, (c) under $(10, 10^{-5})$-DP without BN layers

is affected in original DP-SGD, thus the sensitivity of the gradient will scale up by B, i.e. the batch size. To make the same DP guarantee hold, we can recalculate the ϵ by replacing the original noise multiplier σ with $\sigma \cdot B$ to account for the gradient sensitivity change in a minibatch. However, the actual noise multiplier is much larger and will undoubtedly destroy the model utility.

Batch normalization (BN) vs. Group normalization (GN): Replacing BN with GN is a common practice for training DP models, e.g. as in Z. Luo et al. (2021). However, unlike the image classification network (Z. Luo et al. 2021) that only takes forward pass, flow generative models are invertible models, and any normalization technique will run into issues when we want a backward pass (e.g. sampling), i.e. there is no mean and variance for unnormalization. Currently, BN layer in flow models tends to register two buffers, i.e. running mean and running variance, to keep track of the batch mean and batch variance in the training, then fix it to generate data in the backward pass. We implement a GN layer for flow models similarly, i.e. registering two buffers to keep track of the group mean and variance in the training (we use 4 groups for each instance). However, during generation, we first try repeating the group mean and variance for the sampled batch, but the generation exhibits no variance, i.e. they all look similar. Then we change the registered buffers to keep track of a pool of group mean and variance, but the generation becomes null. We leave a tailored normalization layer that is suitable for private training for future work.

Limitation: The insight behind DP-LFlow, i.e. shrinking the model size, seems to contradict the intuition and the findings in a prior study on scaling laws of language models (Kaplan et al. 2020), because more model parameters generally mean better learning capability. However, Kaplan et al. (2020) conducted all experiments in the non-private setting, and our findings in Figure 3.1 on image generators are consistent with Kaplan et al. (2020) in the non-private setting. In the private setting, Figure 3.1 motivates us to improve the performance of DPGMs by preventing the model from being over-complicated. Nevertheless, it is apparent that an over-simple generative model (certainly including DP-LFlow) is not able to generate realistic images either with or without DP guarantee. Therefore, shrinking the model size may impose an upper bound on the model utility.

3.7 Related Work

We categorize related works by approaches:

DP-SGD: The vast majority of related works are realized by training different generative models with DP-SGD. **GAN:** DP-GAN (Xie et al. 2018) first trains GAN with DP-SGD algorithm, where the discriminator is trained with DP-SGD, then the generator is automatically DP as ensured by post-processing theorem. DP-CGAN (Torkzadehmahani et al. 2019) extends the idea into the conditional setting. **VAE:** DP-VaeGM (Q. Chen et al. 2018) trains k VAEs on k classes of private data with the DP-SGD algorithm, and returns the union as generations. This work only focuses on privacy attacks. DP-kVAE (Acs et al. 2018) first partitions the dataset into k clusters by kernel k-means method, then trains k VAEs on each data cluster with DP-SGD. However, their generation exhibits clear mode-collapse. **Flow:** DP-NF (Waites and Cummings 2021) directly trains a flow-based model by DP-SGD. DP-HFlow (Lee et al. 2022) designs a fine-grained gradient clipping strategy to increase the signal-to-noise ratio and accelerate the training. However, both works relating to flow models are limited to (low dimensional) tabular datasets. **Diffusion model:** A few recent papers also tried training the powerful diffusion models with DP-SGD (Dockhorn et al. 2023; Ghalebikesabi et al. 2023), which can generate realistic images under DP constraints. However, their training is significantly more computation-intensive, e.g. Dockhorn et al. (2023) requires 8 GPUs and one day to train a DP diffusion model on MNIST and FMNIST.

PATE Mechanism: Private Aggregation of Teacher Ensembles (PATE) (Papernot, Abadi, et al. 2017; Papernot, S. Song, et al. 2018) is another mechanism for learning a DP model, by perturbing the aggregated information from an ensemble of teacher models with noise. PATE-GAN (Jordon et al. 2019) first applies PATE mechanism to GAN, where the discriminator becomes non-differentiable, thus a student discriminator is trained with all teacher ensembles, which is then used to train the generator. G-PATE (Long et al. 2021) sanitizes

the aggregated gradients from teacher discriminators to the generator to make the generator DP. However, gradient vectors need to be discretized in each dimension to employ the PATE mechanism that only takes categorical data as input. DataLens (B. Wang, Wu, et al. 2021) further improves G-PATE by introducing a three-step gradient compression and aggregation algorithm called TopAgg.

Kernel-based Methods: DP-MERF (Harder et al. 2021) proposes to perturb the kernel mean embeddings of real data through random Fourier features, and train a generator by minimizing the maximum mean discrepancy (MMD) between the noisy embedding of private input and embedding of generation. PEARL (Liew et al. 2022) extends the idea of DP-MERF by introducing an adversarial objective on sampling frequencies, which indicates improvement in performance.

Others: GS-WGAN (D. Chen et al. 2020) creates a privacy barrier at the output of the generator of a Wasserstein GAN (WGAN), based on the observation that only the generator will be published, thus only the generator needs to be private. DP-Sinkhorn (T. Cao et al. 2021) adds a privacy barrier in a similar way as GS-WGAN, where a Sinkhorn loss is used as the training objective.

3.8 Summary

Though DP-SGD is currently the workhorse algorithm for training a deep learning model, it remains a big challenge whether it can be applied to training large models with acceptable model utility. In this paper, we first show that training a DP flow via DP-SGD is highly challenging (or even infeasible) with achieving acceptable utility due to a few particular challenges of flow models, and then propose an effective solution, i.e. DP-LFlow, by reducing the flow training from the full input space to a lower dimensional latent space, so that the model is more resilient to (larger) noise perturbation introduced by DP-SGD. Experimental results on widely compared image benchmarks demonstrate the generality and scalability of DP-LFlow on different image spaces (grayscale and RGB) and different DP constraints (weak and strong DP guarantees). Notably, to our best knowledge, DP-LFlow is the first DPGM to scale to high-resolution image datasets, which further validates its effectiveness and versatility.

Chapter 4

Functional Rényi Differential Privacy for Generative Modeling

4.1 Preface

Although training a generative model with DP-SGD is the most straightforward way to obtain a DPGM, it suffers from (1) heavy training time and memory overhead, due to the individual gradient clipping, and (2) limited scalability to stronger DP guarantees.

An alternative way to train a DPGM is the kernel method, by making the training objective differentially private, which does not suffer from the same issues as DP-SGD. However, existing kernel-based methods tend to generate homogeneous images, which restricts their utility in practice. In this work, we first extend RDP from vectors to functions, then we apply the proposed functional RDP mechanism to the loss function (in an RKHS) differentially private without trimming the RKHS. The proposed method is more scalable on strict DP guarantees and can generate data with better utility given the same privacy budgets.

Abstract: Differential privacy (DP) has emerged as a rigorous notion to quantify data privacy. Subsequently, Rényi differential privacy (RDP) has become an alternative to the ordinary DP notion in both theoretical and empirical studies, because of its convenient compositional rules and flexibility. However, most mechanisms with DP (RDP) guarantees are essentially based on randomizing a fixed, finite-dimensional vector output. In this work, following Hall et al. (2013) we further extend RDP to functional outputs, where the output space can be infinite-dimensional, and develop all necessary tools, e.g. (subsampled) Gaussian mechanism, composition, and post-processing rules, to facilitate its practical adoption. As an illustration, we apply functional RDP (f-RDP) to functions in the reproducing kernel Hilbert space (RKHS) to develop a differentially private generative model (DPGM), where training can be interpreted as iteratively releasing loss functions (in an RKHS) with DP

guarantees. Empirically, the new training paradigm achieves a significant improvement in privacy-utility trade-off compared to existing alternatives, especially when $\epsilon = 0.2$. Our code is available at `https://github.com/dihjiang/DP-kernel`.

The following sections are adapted from the main paper and appendix of D. Jiang, S. Sun, and Y. Yu (2023). "Functional Renyi Differential Privacy for Generative Modeling". *Advances in Neural Information Processing Systems*, vol. 36.

4.2 Introduction

Modern machine learning has achieved impressive success thanks to the availability of big data and computing resources. However, there are increasing privacy concerns when training with personal or sensitive data. Differential privacy (DP, Dwork 2006) has become the de-facto standard technique for releasing statistics of sensitive databases, which is designed to bound the output change of a randomized mechanism $\mathcal{M}$ given an incremental input deviation, such that $\mathcal{M}$ does not depend too much on any individual point in the dataset. Recently, Mironov (2017) generalizes DP to Rényi differential privacy (RDP) through α-Rényi divergence (Rényi 1961), which shares many properties with the ordinary DP, yet with tighter and easier composition analysis, thereby attracts more attention in practice.

The popular mechanisms (e.g. Gaussian or Laplace) towards DP or RDP essentially randomize a finite-dimensional vector output with (Gaussian or Laplace) noises. However, if we want to privately release a *function*, vector-based DP mechanisms are not readily amenable, because a function over a real-valued domain is characterized by infinitely many points (Hall et al. 2013), which will lead to ∞ bound on the L2/L1-sensitivity if we follow the same sensitivity analysis in Dwork, Roth, et al. (2014) for the vector output, and will effectively lead to infinitely large Gaussian/Laplacian noise and is impractical. Moreover, it is unclear how to add noise to an infinite-dimensional output. Examples that require a functional DP mechanism include privately releasing the reward function in reinforcement learning (B. Wang and Hegde 2019), the kernel function in kernel density estimation and kernel support vector machine (SVM) (Hall et al. 2013), and the kernel function in DPGM in this work.

Hall et al. (2013) made the most fundamental contribution to extending DP from vectors to functions. Essentially, the functional Gaussian mechanism is achieved by adding a sample path of Gaussian process to a function, in contrast to adding Gaussian noise to a vector. Evaluating the released DP function at arbitrarily many points (which will form a vector) will retain the same DP guarantee. It is worth mentioning that there are no composition theorems and subsampled Gaussian mechanisms developed for functional DP in Hall et al. (2013), thus restricting its use in deep learning.

Due to the theoretical convenience and practical flexibility of RDP, in this work, we aim to extend RDP to functions, with all necessary tools to facilitate its adoption in deep

learning. Furthermore, we demonstrate its value via a particular application in DPGM where the loss function is in an RKHS. Our contributions can be summarized as:

- Theoretically, we develop the functional RDP, which is equipped with many useful tools including (subsampled) Gaussian mechanism, composition, and post-processing theorems. We will show that functional RDP will share many important properties and results with the vector-based variant.
- Empirically, with functional RDP, we propose a novel DPGM training paradigm by privatizing the loss function in an RKHS, rather than truncating the RKHS to a finite-dimensional space and injecting Gaussian noise therein as in existing works. Our method is evaluated and compared across a wide variety of image datasets and DP guarantees, where our method consistently outperforms other baselines by a large margin. Notably, our method indicates better scalability at more stringent DP guarantees (e.g., $\epsilon = 1$ and 0.2), compared to state-of-the-art (SoTA) baselines.

4.3 Preliminary

Differential privacy quantifies and restricts the output change of a randomized mechanism given an incremental change in the input dataset, such that the privacy of an individual point is protected. The output of a randomized mechanism can be either a vector (Sections 4.3.1 to 4.3.3) or a function (Section 4.3.4). In this section, we recall a few important related works in differential privacy.

4.3.1 Differential Privacy for Vectors (v-DP)

Definition 4 ((ϵ, δ)-DP for vectors, Dwork 2006; Dwork, Roth, et al. 2014)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $D, D' \in \mathcal{D}$ and for any (measurable) subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta,$$

where adjacent inputs (a.k.a. neighbouring datasets) D, D' only differ in one entry. Particularly, when $\delta = 0$, we say that $\mathcal{M}$ is ϵ-DP.

4.3.2 Rényi Differential Privacy for Vectors (v-RDP)

Mironov (2017) first formalizes Rényi differential privacy (RDP) which extends ordinary DP by using α-Rényi divergence (Rényi 1961). RDP is shown to provide easier composition properties than the ordinary DP notion, and it can be easily converted to (ϵ, δ)-DP.

Definition 5 ((α, ϵ)-RDP for vectors, Mironov 2017). *A randomised mechanism $\mathcal{M}$ is (α, ϵ)-RDP if for all adjacent inputs D, D', Rényi's α-divergence (of order $\alpha > 1$) between the distributions of $\mathcal{M}(D)$ and $\mathcal{M}(D')$ satisfies:*

$$\mathbb{D}_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) := \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim q} \left(\frac{p(x)}{q(x)} \right)^\alpha \leq \epsilon,$$

where p and q are the density of $\mathcal{M}(D)$ and $\mathcal{M}(D')$, respectively.

Conveniently, RDP is linearly composable:

Theorem 6 (Sequential composition of v-RDP, Mironov 2017). *Let $f : \mathcal{D} \to \mathcal{R}_1$ be (α, ϵ_1)-RDP, $g : \mathcal{R}_1 \times \mathcal{D} \to \mathcal{R}_2$ be (α, ϵ_2)-RDP, then running f, g sequentially to obtain $h : \mathcal{D} \to \mathcal{R}_1 \times \mathcal{R}_2, h(D) := \big(f(D), g(f(D), D)\big)$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Similar to the parallel composition theorem for ordinary ϵ-DP as in McSherry (2009), we complement the parallel composition for v-RDP:

Theorem 7 (Parallel composition of v-RDP). *If mechanism $\mathcal{M}_i$ satisfies (α, ϵ_i)-RDP for $i = 1, 2, \ldots, m$, and let $D_1, D_2, \ldots, D_m$ be the disjoint partitions by executing a deterministic partitioning function P on D. Releasing $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_m(D_m)$ satisfies $(\alpha, \max_{i \in I} \epsilon_i)$-RDP, where $I = \{1, 2, \ldots, m\}$.*

4.3.3 Gaussian Mechanism for v-DP and v-RDP

Among multiple choices, the Gaussian mechanism is more suitable for the (ϵ, δ)-DP notion (where $\delta > 0$) and provides additional flexibility (e.g., the sum of Gaussians is still a Gaussian). It is achieved by adding calibrated spherical Gaussian noise to a vector output.

Proposition 1 (Gaussian mechanism for v-DP and v-RDP, Dwork, Roth, et al. 2014; Mironov 2017). *Given a d-dimensional function $f : \mathcal{D} \to \mathbb{R}^d$. The Gaussian mechanism is given by:*

$$\mathcal{M}(D) = f(D) + \sigma \Delta_2 f \cdot \mathcal{N}(0, \mathbb{I}_d),$$

where $\Delta_2 f = \max_{D \sim D', D, D' \in \mathcal{D}} \|f(D) - f(D')\|_2$. $\mathcal{M}(D)$ is said to be: (1) (ϵ, δ)-DP if $\sigma \geq \sqrt{2 \ln(1.25/\delta)}/\epsilon$ for $\epsilon \in (0, 1)$, or (2) $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP.

4.3.4 Differential Privacy for Functions (f-DP)

Prior DP works mainly focused on the setting when the output of a query is a vector. However, if the output is a *function*, such as the kernel density estimation example in Hall et al. (2013) and the DPGM example in our work (Section 4.5.3), a few challenges render

the direct application of previous DP results on a *functional output*[1] intractable: it is unclear (1) how to analyze the sensitivity and (2) how to add Gaussian noise to an infinite-dimensional output. To the best of our knowledge, Hall et al. (2013) are the pioneers in addressing those challenges by starting with defining the (ϵ, δ)-DP for functions:

Definition 6 ((ϵ, δ)-DP for functions, Hall et al. 2013)**.** *Consider a class of functions indexed by database D over $T = \mathbb{R}^d$, i.e. $\{f_D : D \in \mathcal{D}\} \subseteq \mathbb{R}^T$. Define cylinder sets*[2] *$C_{S,B} = \{f \in \mathbb{R}^T : (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) \in B\}$, for all finite subsets $S = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of T and Borel sets*[3] *$B \in \mathbb{R}^n$. Then, define $\mathcal{C}_s = \{C_{S,B} : B \in \mathcal{B}(\mathbb{R}^n)\}$ and $\mathcal{F}_0 = \bigcup_{S:|S|<\infty} \mathcal{C}_s$. We say the mechanism $\widetilde{f_D}$ satisfies (ϵ, δ)-DP over the field of cylinder sets, if for all $D, D' \in \mathcal{D}$:*

$$\Pr[\widetilde{f_D} \in A] \leq \exp(\epsilon) \times \Pr[\widetilde{f_{D'}} \in A] + \delta, \quad \forall A \in \mathcal{F}_0. \tag{4.1}$$

Hall et al. (2013) point out that whenever Eq. 4.1 holds, for any finite set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in T chosen a-priori, or after the construction of the function, or even adaptively chosen based on the outputs given, releasing the vector $[\widetilde{f_D}(\mathbf{x}_1), \ldots, \widetilde{f_D}(\mathbf{x}_n)]$ satisfies (ϵ, δ)-DP.

The Gaussian mechanism for f-DP [4] is reached by injecting a sample path of a calibrated Gaussian process[5] into the function:

Proposition 2 (Gaussian mechanism for f-DP, Hall et al. 2013)**.** *Let G be a sample path of a Gaussian process having mean zero and covariance function k. Let M denote the Gram matrix*

$$M(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}. \tag{4.2}$$

Let $\{f_D : D \in \mathcal{D}\}$ be a family of functions indexed by database D. Releasing $\widetilde{f_D} = f_D + \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon \cdot G$ satisfies (ϵ, δ)-DP whenever

$$\sup_{D \sim D'} \sup_{n<\infty} \sup_{(\mathbf{x}_1,\ldots,\mathbf{x}_n) \in T^n} \left\| M^{-\frac{1}{2}}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \begin{pmatrix} f_D(\mathbf{x}_1) - f_{D'}(\mathbf{x}_1) \\ \vdots \\ f_D(\mathbf{x}_n) - f_{D'}(\mathbf{x}_n) \end{pmatrix} \right\|_2 \leq \Delta. \tag{4.3}$$

[1] It means the output is a function.

[2] Intuitively, the cylinder set defined here is a set of functions where any finite evaluation of the function is in a "ball".

[3] A Borel set is formed via countable union, intersection and complement of open sets in a topological space.

[4] Not to confuse with the function DP in Dong et al. (2019), where the type-I and type-II errors are controlled by a pre-specified function.

[5] A Gaussian process is a generalization of the multivariate Gaussian distribution to the infinite-dimensional case. Any arbitrarily collected finite random variables from a Gaussian process follow a multivariate Gaussian distribution.

Particularly, Hall et al. (2013) studied how to achieve (ϵ, δ)-DP for functions in an RKHS [6] $\mathcal{H}$:

Corollary 1 (Corollary 9 in Hall et al. 2013). *For $\{f_D : D \in \mathcal{D}\} \subseteq \mathcal{H}$, releasing $\widetilde{f_D} = f_D + \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon \cdot G$ is (ϵ, δ)-DP (with respect to the cylinder σ-field) whenever $\sup_{D,D'} \|f_D - f_{D'}\|_{\mathcal{H}} \leq \Delta$ and when G is a sample path of a Gaussian process with mean zero and covariance function k that is given by the reproducing kernel of $\mathcal{H}$.*

4.4 Rényi Differential Privacy for Functions (f-RDP)

In this section, we aim to extend the definition of RDP to functional outputs, along with its associated calculus rules to facilitate practical adoption. In particular, we will show that the main results for v-RDP all extend to f-RDP.

4.4.1 Definition

Consider a class of functions over $T = \mathbb{R}^d$, i.e. $\{f_D : D \in \mathcal{D}\} \subseteq \mathbb{R}^T$. We can define a functional generalization of the α-Rényi divergence:

Definition 7 ((α, ϵ)-RDP for functions). *Denote the evaluation of function $\widetilde{f_D}$ on any finite subsets $S = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of T by $\{\widetilde{f_D}(\mathbf{x}_1), \ldots, \widetilde{f_D}(\mathbf{x}_n)\} := \widetilde{f_D}(S)$. We say $\widetilde{f_D}$ is (α, ϵ)-RDP, if for all adjacent inputs $D, D' \in \mathcal{D}$, Rényi's α-divergence (of order $\alpha > 1$) between the distributions of $\widetilde{f_D}(S)$ and $\widetilde{f_{D'}}(S)$ satisfies:*

$$\mathbb{D}_\alpha\big(\widetilde{f_D}(S) \| \widetilde{f_{D'}}(S)\big) := \tfrac{1}{\alpha-1} \log \mathbb{E}_{x\sim q} \left(\tfrac{p(x)}{q(x)}\right)^\alpha \leq \epsilon, \tag{4.4}$$

where p, q are the density of $\widetilde{f_D}(S)$ and $\widetilde{f_{D'}}(S)$, respectively.

Remark 1. *Definition 7 essentially claims that the distribution of any finite number of evaluations of function $\widetilde{f_D}$ and $\widetilde{f_{D'}}$ satisfies Definition 5 (v-RDP).*

We have a weaker definition of f-RDP (see Lemma 1 for the *weaker* definition) also based on cylinder sets as in Definition 6. We also refer interested readers to Appendix C.6 for an alternative unified definition of RDP for both vectors and functions.

4.4.2 Post-processing Theorem

As any data-independent post-processing preserves the DP guarantee for v-RDP (Definition 5), it also preserves DP guarantee for f-RDP with Remark 1. Specifically,

[6] A Hilbert space of functions with continuous pointwise evaluation $f \mapsto f(x)$. We give more explanations in Section 4.5.1.

Theorem 8 (Post-processing theorem of f-RDP). *If a function f_D is (α, ϵ)-RDP, so is $g \circ f_D$, where g is a post-processing mechanism that only depends on a finite number of outputs of f_D.*

4.4.3 Conversion to (ϵ, δ)-DP

The following conversion is a direct extension of its vector version in Mironov (2017):

Proposition 3 (f-RDP conversion to f-DP). *A function $\widetilde{f_D}$ that is (α, ϵ)-RDP is $(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta)$-DP.*

4.4.4 Composition Theorems

Similar to Theorem 7, we first derive the parallel composition theorem of RDP for functions:

Theorem 9 (Parallel composition of f-RDP). *Given a deterministic partitioning function P, let $D_1, D_2, \ldots, D_m$ be the disjoint partitions by executing P on D. If function f_{D_i} satisfies (α, ϵ_i)-RDP for $i = 1, 2, \ldots, m$, releasing $(f_{D_1}, \ldots, f_{D_m}) := f_D$ satisfies $(\alpha, \max_{i \in I} \epsilon_i)$-RDP, where $I = \{1, 2, \ldots, m\}$.*

Now we move on to the sequential composition theorem of RDP for functions (extension of Theorem 6 to functional mechanisms), which is important and required for composing the total privacy cost when we sample different sample paths from the Gaussian process over training iterations.

First, we note that Theorem 6 implicitly defines g as a functional mechanism, i.e. $g : \mathcal{D} \to \mathcal{R}_2^{\mathcal{R}_1}$. It means for any $r_1 \in \mathcal{R}_1$, we have $g_{r_1} := g(r_1, \cdot) : \mathcal{D} \to \mathcal{R}_2$ is (α, ϵ_2)-RDP.

Theorem 10 (Sequential composition of f-RDP). *Let $\{f_D : D \in \mathcal{D}\}$ and $\{g_D : D \in \mathcal{D}\}$ be two families of functions indexed by dataset D, where $f_D \in \mathcal{R}_1^T$ is (α, ϵ_1)-RDP and $g_D : \mathcal{R}_1^T \to \mathcal{R}_2^S$ is (α, ϵ_2)-RDP. Releasing the sequentially composed functional mechanism $h_D = (f_D, g_D \circ f_D) \in \mathcal{R}_1^T \times \mathcal{R}_2^S = (\mathcal{R}_1 \times \mathcal{R}_2)^{T \times S}$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

4.4.5 Gaussian Mechanism

We also need to develop a Gaussian mechanism to retain f-RDP. To make it more convenient, we first rewrite Proposition 1 in a similar form to Proposition 3 in Hall et al. (2013) as:

Definition 8 (Gaussian mechanism for v-RDP, with non-isotropic Gaussian). *Let $M \in \mathbb{R}^{d \times d}$ be a positive definite symmetric matrix, the family of vectors $\{\mathbf{v}_D : D \in \mathcal{D}\} \in$*

$\mathbb{R}^d$ *satisfies* $\sup_{D\sim D'} \|M^{-\frac{1}{2}}(\mathbf{v}_D - \mathbf{v}_{D'})\|_2 \leq \Delta$ *for all adjacent datasets* $D, D' \in \mathcal{D}$. *The Gaussian mechanism is given by:*

$$\widetilde{\mathbf{v}_D} = \mathbf{v}_D + \sigma\Delta \cdot \mathcal{N}(0, M). \tag{4.5}$$

$\widetilde{\mathbf{v}_D}$ in Eq. 4.5 satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP. Now we define the Gaussian mechanism for f-RDP:

Proposition 4 (Gaussian mechanism for f-RDP)**.** *Let* G *be a sample path of a Gaussian process having mean zero and covariance function* k. *Let* M *denote the Gram matrix (as defined in Eq. 4.2). Let* $\{f_D : D \in \mathcal{D}\}$ *be a family of functions indexed by database* D. *Releasing* $\widetilde{f_D} = f_D + \sigma\Delta \cdot G$ *satisfies* $(\alpha, \frac{\alpha}{2\sigma^2})$*-RDP whenever Eq. 4.3 holds.*

Particularly, when the function is in an RKHS $\mathcal{H}$, we have the following corollary:

Corollary 2. *For* $\{f_D : D \in \mathcal{D}\} \subseteq \mathcal{H}$, *releasing* $\widetilde{f_D} = f_D + \sigma\Delta \cdot G$ *is* $(\alpha, \frac{\alpha}{2\sigma^2})$*-RDP (with respect to the cylinder* σ*-field) whenever* $\sup_{D,D'} \|f_D - f_{D'}\|_{\mathcal{H}} \leq \Delta$ *and when* G *is a sample path of a Gaussian process with mean zero and covariance function* k *given by the reproducing kernel of* $\mathcal{H}$.

4.4.6 Subsampled Gaussian Mechanism (SGM)

Subsampling is a crucial component in existing DP deep learning algorithms (e.g., DP-SGD of Abadi et al. 2016), which incurs a small privacy cost for every sampled batch by querying D, thereby requiring to compose the total privacy loss (measured by ϵ) over training iterations. The same thing could happen for functional mechanisms when we subsample a batch S from the whole dataset D to index the function f_S.

The current dominant (python-based) DP implementation packages, e.g., Tensorflow privacy and Opacus, apply v-RDP and compute the total ϵ in three main steps: (1) compute the v-RDP guarantee for an SGM, based on a numerical procedure in Mironov et al. (2019); (2) sequentially compose v-RDP over training iterations; (3) convert v-RDP to v-DP. In previous sections we have already shown that f-RDP shares the same results with v-RDP on steps (2) and (3), now we will show that an SGM for f-RDP can be reduced to v-RDP in Mironov et al. (2019), thus, the numerical procedure in Mironov et al. (2019) (and thereby those DP packages) is amenable to f-RDP in terms of the ϵ accumulation.

We apply the same subsampling strategy as in Mironov et al. (2019). Let S be a subsampled set from D, where each element of S is independently drawn from D with probability q. SGM for f-RDP is given by $\widetilde{f_S} = f_S + \sigma\Delta \cdot G$, where Δ and G are defined in Proposition 4. The reduction is immediate: for any finite set of points $V = \{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ (where $d < \infty$), $f_S(V)$ will form a d-dimensional vector. Let $g_S(V) = M^{-1/2} f_S(V)$, we have $\widetilde{g_S}(V) = g_S(V) + \sigma\Delta \cdot \mathcal{N}(0, \mathbb{I}_d)$, where $g_S(V)$ is a d-dimensional vector and $\widetilde{g_S}(V)$ is the same SGM as in Mironov et al. (2019). Therefore, f-RDP shares the same guarantee of

an SGM with v-RDP. Another intuition is that Mironov et al. (2019) reduce computing the α-Rényi divergence of d-dimensional Gaussians to 1-dimensional Gaussians, which guides all of their subsequent derivations. When $d \to \infty$, we can reach the same reduction from infinite-dimensional Gaussian (Gaussian process) to 1-dimensional Gaussian.

4.5 An Application in DPGM

To demonstrate the practical value of f-RDP, here we consider a particular example of training a differentially private generative model (DPGM) through Maximum Mean Discrepancy (MMD).

4.5.1 Background

Assuming a feature map $\phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is an RKHS of some kernel, MMD (Gretton et al. 2012) is a non-parametric distance measure that compares two distributions p, q by:

$$\mathcal{L}_{\mathrm{MMD}^2}(p, q) = \|\mathbb{E}_{\mathbf{x}\sim p}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{w}\sim q}[\phi(\mathbf{w})]\|_{\mathcal{H}}^2,$$

where $\mathbb{E}_{\mathbf{x}\sim p}[\phi(\mathbf{x})] \in \mathcal{H}$ is also known as the (kernel) mean embedding (KME) of p.

Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that[7] $k(\mathbf{x}, \mathbf{w}) = \langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle_{\mathcal{H}}$, we can play the kernel trick to compute the (squared) MMD in an alternative way:

$$\mathcal{L}_{\mathrm{MMD}^2}(p, q) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim p} k(\mathbf{x}, \mathbf{x}') - 2\mathbb{E}_{\mathbf{x}\sim p,\mathbf{w}\sim q} k(\mathbf{x}, \mathbf{w}) + \mathbb{E}_{\mathbf{w},\mathbf{w}'\sim q} k(\mathbf{w}, \mathbf{w}'),$$

which also implicitly lifts the KME into an infinite-dimensional space.

Sriperumbudur et al. (2011) suggest that if k is a characteristic kernel (e.g., Gaussian kernel), then MMD $= 0$ iff $p = q$, which makes MMD a practical tool in many applications, such as two-sample test (Gretton et al. 2012) and generative modeling (e.g., C.-L. Li et al. 2017; Y. Li et al. 2015).

4.5.2 Related Works

Balog et al. (2018) first proposed a DP database release mechanism via KME, by truncating the infinite-dimensional RKHS to a finite-dimensional feature space through random Fourier features and adding Gaussian noise to the mean of truncated feature embeddings of all data. This idea is further extended to generative modeling by DP-MERF (Harder et al. 2021). Specifically, given the samples drawn from two distributions: $D = \{\mathbf{x}_i\}_{i=1}^N \sim p$

[7] By Moore–Aronszajn theorem, any symmetric, positive definite kernel k on $\mathcal{X}$ defines a unique Hilbert space of functions on $\mathcal{X}$ where k is a reproducing kernel. We can find a feature map $\phi : \mathcal{X} \to \mathcal{H}$, such that $k(\mathbf{x}, \mathbf{w}) = \langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle_{\mathcal{H}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product on $\mathcal{H}$.

(true) and $W = \{\mathbf{w}_j\}_{j=1}^M \sim q$ (generated), empirical MMD with a kernel function k can be estimated by:

$$\widehat{\mathcal{L}_{\mathrm{MMD}^2}}(p,q) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{p=1}^{N} k(\mathbf{x}_i,\mathbf{x}_p) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} k(\mathbf{x}_i,\mathbf{w}_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{q=1}^{M} k(\mathbf{w}_j,\mathbf{w}_q). \tag{4.6}$$

DP-MERF approximates the kernel by: $k(\mathbf{x},\mathbf{w}) = \widehat{\phi}(\mathbf{x})^\top \widehat{\phi}(\mathbf{w})$, where $\widehat{\phi}(\mathbf{x}) \in \mathbb{R}^d$ is the random Fourier feature (Rahimi and Recht 2007) and d is the feature dimension. Now the loss becomes:

$$\widehat{\mathcal{L}_{\mathrm{MMD}^2_{rf}}}(p,q) = \left\| \frac{1}{N}\sum_{i=1}^{N}\widehat{\phi}(\mathbf{x}_i) - \frac{1}{M}\sum_{j=1}^{M}\widehat{\phi}(\mathbf{w}_j)\right\|_2^2 = \left\|\widehat{\mu}_p - \frac{1}{M}\sum_{j=1}^{M}\widehat{\phi}(\mathbf{w}_j)\right\|_2^2,$$

where $\widehat{\mu}_p = \frac{1}{N}\sum_{i=1}^{N}\widehat{\phi}(\mathbf{x}_i)$ is the empirical KME of p. Calibrated Gaussian noise is added to $\widehat{\mu}_p$ to obtain $\widetilde{\mu}_p$ with DP guarantees, which can be viewed as privatized statistics of the whole real dataset. Thereafter, the training objective is to match the KME of generated data with $\widetilde{\mu}_p$, without querying the real data any longer. A line of recent follow-up works, e.g., PEARL (Liew et al. 2022) and DP-HP (Vinaroz et al. 2022), boil down to improving the finite-dimensional truncation and show some further improvement in utility.

Compared to other DPGM approaches via DP-SGD (Abadi et al. 2016), DP-MERF is appealing in two aspects: (1) training efficiency, since noise is added to the KME of the whole database once and for all, whereas DP-SGD has to clip and perturb the gradient in each training iteration, which leads to significant training time overhead; (2) more scalable to smaller ϵ, e.g., $\epsilon = 1$ or below.

However, we observe that the generation by DP-MERF (and related followup methods) resembles "mean-like" images (e.g. see Figures 4.1 to 4.3), which can be explained by their training objective, because matching the KME of all data is likely to lead to *mode collapse*. Moreover, truncating the RKHS into a finite-dimensional space makes it easy to add Gaussian noise, but at the cost of potentially losing the fine ability to distinguish the data distribution from generation (any finite-dimensional RKHS is not characteristic). Therefore, we turn to study the possibility of adding noise in the infinite-dimensional RKHS directly.

4.5.3 Methodology

We use Gaussian kernel as our kernel function k, i.e., $k(\mathbf{x},\mathbf{w}) = \exp\{-\frac{\|\mathbf{x}-\mathbf{w}\|_2^2}{2h^2}\}$, since it is widely used in related works (Harder et al. 2021; Liew et al. 2022; Vinaroz et al. 2022). Define

$$f_D = \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}_i) = \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i,\cdot),$$

so we can rewrite Eq. 4.6 by plugging in f_D:

$$\widehat{\mathcal{L}_{\text{MMD}^2}} = \frac{1}{N}\sum_{p=1}^{N} f_D(\mathbf{x}_p) - \frac{2}{M}\sum_{j=1}^{M} f_D(\mathbf{w}_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{q=1}^{M} k(\mathbf{w}_j, \mathbf{w}_q).$$

Privatizing the terms relating to real data $\mathbf{x}_i \in D$ $(i = 1, 2, \ldots, N)$ amounts to privatizing the function $f_D \in \mathcal{H}$ by Corollary 2, which leads to our private training objective:

$$\widetilde{\mathcal{L}_{\text{MMD}^2}} = \frac{1}{N}\sum_{p=1}^{N} \widetilde{f_D}(\mathbf{x}_p) - \frac{2}{M}\sum_{j=1}^{M} \widetilde{f_D}(\mathbf{w}_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{q=1}^{M} k(\mathbf{w}_j, \mathbf{w}_q), \tag{4.7}$$

where $\mathbf{w} = g_\theta(\mathbf{z})$, i.e., $\mathbf{w}$ is generated from standard Gaussian noise $\mathbf{z}$ by a generative neural network g_θ. Given the kernel is Gaussian, we can easily bound the sensitivity of f_D in the RKHS norm. Without loss of generality, assume D, D' only differ in the last element, i.e., $\mathbf{x}_N \neq \mathbf{x}'_N$. Then,

$$(f_D - f_{D'}) = \frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^{N} \phi(\mathbf{x}'_i) = \frac{1}{N}\big(\phi(\mathbf{x}_N) - \phi(\mathbf{x}'_N)\big).$$

Now we play the kernel trick $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}$ again:

$$\|f_D - f_{D'}\|_{\mathcal{H}}^2 = \frac{1}{N^2}\big(k(\mathbf{x}_N, \mathbf{x}_N) - 2k(\mathbf{x}_N, \mathbf{x}'_N) + k(\mathbf{x}'_N, \mathbf{x}'_N)\big) \leq \frac{2}{N^2} := \Delta^2.$$

We follow *the batch method* in Hall et al. (2013) to release function $\widetilde{f_D}$ in practice, as it naturally fits the batch training manner, which amounts to sampling a path from the Gaussian process specified by any finite collection (batch) of points. Assuming the batch size is m, we concatenate $(\mathbf{x}, \mathbf{w}) = \mathbf{s}$ (of size $2m$, $\mathbf{x} = \mathbf{s}_{[1:m]}, \mathbf{w} = \mathbf{s}_{[m+1:2m]}$), to save an additional privacy cost incurred by an additional sample path in each training iteration. Now we release $\widetilde{f_D}(\mathbf{x}_i)$ and $\widetilde{f_D}(\mathbf{w}_j)$ via $\widetilde{f_D}(\mathbf{s})$ as needed in Eq. 4.7 by:

$$\begin{pmatrix} \widetilde{f_D}(\mathbf{s}_1) \\ \vdots \\ \widetilde{f_D}(\mathbf{s}_{2m}) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} f_D(\mathbf{s}_1) \\ \vdots \\ f_D(\mathbf{s}_{2m}) \end{pmatrix}, \sigma\Delta \begin{pmatrix} k(\mathbf{s}_1, \mathbf{s}_1) & \ldots & k(\mathbf{s}_1, \mathbf{s}_{2m}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{s}_{2m}, \mathbf{s}_1) & \ldots & k(\mathbf{s}_{2m}, \mathbf{s}_{2m}) \end{pmatrix} \right). \tag{4.8}$$

In summary, each training iteration proceeds with (1) subsample a batch S from D to index the function f_S; (2) perturb f_S by Corollary 2; (3) compute the loss function Eq. 4.7 by Eq. 4.8.

Extension to the conditional setting: We follow the approach in DP-MERF to encode labels in the MMD loss. Consider a new kernel k^* as a product of two existing kernels: $k^*\big((\mathbf{x}, \mathbf{y}), (\tilde{\mathbf{x}}, \tilde{\mathbf{y}})\big) = k_{\mathbf{x}}(\mathbf{x}, \tilde{\mathbf{x}}) k_{\mathbf{y}}(\mathbf{y}, \tilde{\mathbf{y}})$, where we set $k_{\mathbf{x}}$ the same as the unconditional setting (i.e., Gaussian kernel) and $k_{\mathbf{y}}$ to be polynomial kernel of order-1, i.e., $k_{\mathbf{y}}(\mathbf{y}, \tilde{\mathbf{y}}) = \mathbf{y}^\top \tilde{\mathbf{y}}$. Now the function that we want to privately release becomes $f_D^* = \frac{1}{N}\sum_{i=1}^{N} k^*((\mathbf{x}_i, \mathbf{y}_i), (\cdot, \cdot))$. The sensitivity of f_D^* is the same as f_D. Thus, releasing f_D^* by *the batch method* is achieved by replacing f_D, k with f_D^*, k^* in Eq. 4.8.

4.5.4 Experiments

Since our method is based on making the RKHS features of a (Gaussian) kernel differentially private without truncating the kernel, we term our method **DP-kernel**. In this section, we will evaluate DP-kernel both qualitatively and quantitatively on three image benchmarks, including both grayscale and colorful images. All implementation details are given in Appendix C.3.

Datasets: We consider widely used image benchmarks in related works, i.e. MNIST (LeCun et al. 1998), Fashion MNIST (Xiao et al. 2017), and CelebA (Liu, P. Luo, et al. 2015). For MNIST and Fashion MNIST, we generate images conditioned on 10 respective labels. For CelebA, we condition on gender. Descriptions and pre-processing of the datasets are given in Appendix C.2.

Evaluation metrics: For quantitative comparison with baselines, we compute the following two metrics via 60k generated images under the same DP guarantees as baselines:

- Generation fidelity. We mainly use Fréchet Inception Distance (FID) (Heusel et al. 2017), as it is widely compared in related works. However, recent studies suggest that FID is not always consistent with human evaluations (e.g. (Stein et al. 2023)), where the authors found that Inception-V3 (the feature extractor used to compute FID, which was proposed in 2015) is not competent enough, and switching to a more recent and powerful feature extractor (e.g. DINOv2 ViT-L/14) seems to overcome this issue. Therefore, we follow Stein et al. (2023) and use their code to compute an additional Fréchet Distance, which is denoted by $\mathrm{FD}_{\mathrm{DINOv2}}$, for our presentation only.
- Generation utility. We train a convolutional neural network (CNN) as the classifier on generated images, then test the classifier on real images, where the performance is measured by the classification accuracy. We take 5 runs and report the average and standard deviation.

Baselines: We compare our method with the following related works: (1) kernel-based methods: DP-MERF (Harder et al. 2021), DP-HP (Vinaroz et al. 2022), PEARL (Liew et al. 2022); (2) others: DP-CGAN (Torkzadehmahani et al. 2019), GS-WGAN (D. Chen et al. 2020), DP-Sinkhorn (T. Cao et al. 2021), G-PATE (Long et al. 2021), DataLens (B. Wang, Wu, et al. 2021), DPDM (Dockhorn et al. 2023). All baselines are developed from v-RDP or v-DP.

Privacy regimes: Note that according to Definition 4, the DP guarantee is weak when $\epsilon \geq 10$, because $\exp(10) \approx 2.2 \times 10^4$, which is meaningless, because the two probabilities are presumed to be comparable for practical deployment (e.g. $\epsilon \leq 1$). However, a line

of recent SoTA DPGMs only generate acceptable images at $\epsilon = 10$ (T. Cao et al. 2021; D. Chen et al. 2020; Torkzadehmahani et al. 2019). Instead, we consider three values of ϵ, i.e., $10, 1, 0.2$, indicating three levels of DP guarantees, where we put comparison under $(10, 10^{-5})$-DP in Appendix C.4 for completeness.

4.5.4.1 Conditional vs. Parallel

The downstream classification task requires access to labels, so we need to consider conditional generative modeling, as introduced at the end of Section 4.5.3.

Alternatively, we can partition the dataset by labels, train K (K is the number of total classes) unconditional sub-models in parallel with DP guarantees, and release the union in the end, which could retain the same level of DP guarantee as the conditional one by parallel composition theorem for f-RDP (Theorem 9). Specifically, we partition the dataset D into K subsets by labels, and set the target ϵ (for f_D) the same for all f_{D_K}. Thus, releasing $\left(f_{D_1}, \ldots, f_{D_K}\right)$ also satisfies (α, ϵ)-RDP. We will show both conditional and parallel generation results in the following sections.

4.5.4.2 Grayscale Image Sets: MNIST and Fashion MNIST

We first tried our method on MNIST and Fashion MNIST. Qualitatively, while all baselines are able to generate reasonable images under $(10, 10^{-5})$-DP guarantee (see Appendix C.4), our method indicates more visual improvements for smaller ϵ with more diversity and less artifacts, as shown in Figures 4.1 and 4.2. The quantitative comparison is summarized in Table 4.1. Although DPDM[8] is the only related work that is on a par with our method (both variants) when $\epsilon = 1$, DP-kernel significantly outperforms other baselines when $\epsilon = 0.2$. Specifically, our method improves the SoTA FID from 61.9 to 26.5 on MNIST, and from 78.4 to 46.2 on FMNIST, while SoTA classification accuracy is generally improved for more than 20% on MNIST and more than 10% on FMNIST.

The $\text{FD}_{\text{DINOv2}}$ of our conditional variant on MNIST and FMNIST are 454.8 and 608.9 when $\epsilon = 1$, and they are 410.3 and 589.2 when $\epsilon = 0.2$.

4.5.4.3 RGB Image Sets: CelebA

To test the versatility and scalability of our method, we also evaluate our method on a more complex colorful image dataset, i.e. CelebA. Figure 4.3 shows that our method generates more diverse face images with identifiable gender attributes compared to baselines, which is quantitatively verified by significant improvement in both FID and Acc in Table 4.1. Note that DPDM is unconditional on CelebA. Notably, for $\epsilon = 0.2$, our method improves the SoTA FID from 264.8 to 85.9, and improves SoTA Acc from 52.7 to 86.5.

[8] DPDM requires 8 GPUs to train, whereas our method only requires 1 GPU.

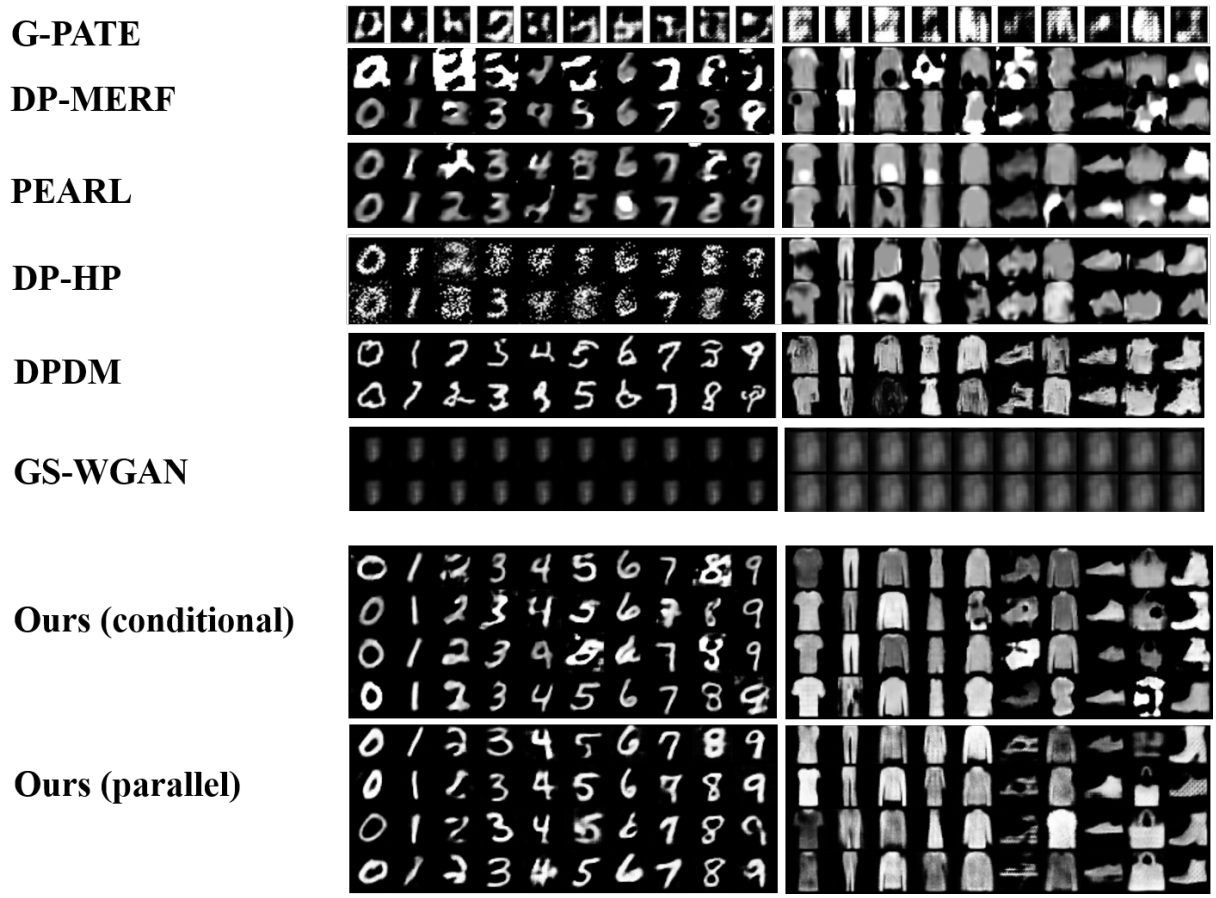


Figure 4.1: Qualitative comparison under $(1, 10^{-5})$-DP on MNIST and Fashion MNIST

The $\text{FD}_{\text{DINOv2}}$ of our conditional variant on CelebA are 1028.2 when $\epsilon = 1$ and 998.9 when $\epsilon = 0.2$.

4.6 Summary

We generalize RDP for vectors to functional mechanisms and develop all building blocks, e.g. (subsampled) Gaussian mechanisms, composition theorems, and post-processing theorems, to facilitate its adoption in deep learning frameworks. We show that those main properties or results of v-RDP also hold for f-RDP. Equipped with f-RDP, we propose a novel approach for training a DPGM, by making the loss function in the RKHS private without truncating the RKHS. Experimental results across different datasets and privacy costs indicate that our method (equipped with f-RDP and retaining the full discriminative capability of the kernel) consistently outperforms other kernel-based methods (with v-RDP) as well as non-kernel-based methods by a large margin. We expect our work to bridge the gap between RDP and functional mechanisms and enrich the family of DPGM.

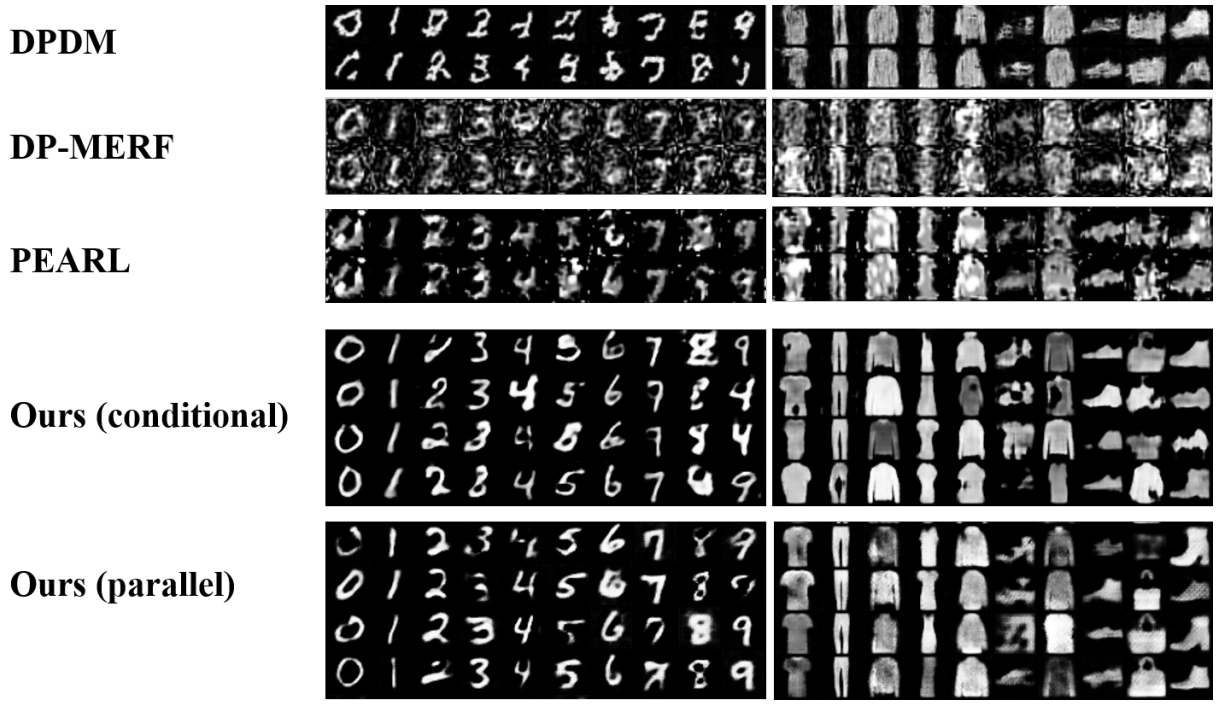


Figure 4.2: Qualitative comparison under $(0.2, 10^{-5})$-DP on MNIST and Fashion MNIST



(a) $\epsilon = 1$ (b) $\epsilon = 0.2$

Figure 4.3: The qualitative comparison on CelebA. $\delta = 10^{-5}$. DPDM is unconditional.

Table 4.1: Quantitative comparison on MNIST, Fashion MNIST (FMNIST) and CelebA. Acc denotes classification accuracy, which is shown in %. ↑ and ↓ refer to higher is better or lower is better, respectively. We use boldface for the best performance. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021) and Long et al. (2021). Results of PEARL and DPDM are cited from Dockhorn et al. (2023). Results of G-PATE and DataLens are cited from their papers, respectively. (*): DPDM did unconditional generation on CelebA.

Method	ϵ	MNIST		FMNIST		CelebA	
		FID ↓	Acc ↑	FID ↓	Acc ↑	FID ↓	Acc ↑
DP-MERF	1	118.3	80.5	104.2	73.1	219.4	57.6
GS-WGAN	1	489.8	14.3	587.3	16.6	437.3	62.9
DP-HP	1	-	74.0	-	67.0	-	-
PEARL	1	121.0	78.2	109.0	68.3	-	-
G-PATE	1	153.4	58.8	214.8	58.1	293.2	70.2
DataLens	1	186.1	71.2	195.0	64.8	297.7	70.6
DPDM	1	23.4	93.4	**37.8**	73.6	*__71.0__	-
Ours (conditional)	1	29.5	93.4±0.5	49.5	78.8±0.4	**81.8**	86.2±0.9
Ours (parallel)	1	**21.8**	**95.5**±0.6	48.4	**80.0**±0.5	83.8	**87.0**±0.8
DP-MERF	0.2	119.3	75.2	151.3	67.4	264.8	52.7
PEARL	0.2	133.0	77.6	160.0	68.0	-	-
G-PATE	0.2	-	22.0	-	18.0	-	-
DataLens	0.2	-	23.4	-	22.3	-	-
DPDM	0.2	61.9	71.9	78.4	57.0	-	-
Ours (conditional)	0.2	**26.5**	91.3±0.8	**46.2**	78.4±0.7	**85.9**	84.3±1.1
Ours (parallel)	0.2	37.3	**93.7**±0.7	65.4	**79.2**±0.5	95.8	**86.5**±1.2

Chapter 5

Conclusion

Despite the impressive success of modern DGMs in generating realistic samples, their trustworthiness remains a significant concern. This thesis examines two critical aspects of trustworthiness—distribution shift and privacy—and their connections to DGMs. DGMs serve roles beyond data generators. For instance, we introduce a reliable out-of-distribution (OOD) detection method using flow generative models (Chapter 2). Besides, both the general public and research communities have increasing concerns about the privacy leakage of DGMs, rather than merely caring about their generation quality. In this thesis, we also introduce two approaches to train a DPGM, which limits the probability of inferring individual data privacy in DGMs (Chapters 3 and 4). Specifically,

- In Chapter 2, although thresholding the likelihood assigned by flow generative models is found to be unreliable in detecting OOD input, we propose a mathematically sound OOD detection method using the bijectivity property of flow models based on a goodness of fit test and random projection. In addition, we adapt our method into a two-sample variant to account for model training insufficiency, making no assumptions on OOD data. Finally, we confirm on larger models and real datasets that model accuracy may be negatively correlated with OOD detection performance.

  It is worth mentioning that the proposed method based on distribution comparison works in a group- or batch-wise manner. Our initial attempt to adapt it into a point-wise method, i.e. creating a group of data by data augmentation and then running the group-wise detection method, fails to achieve ideal performance. In the future, a more effective point-wise detection method is worth exploring. We may also need to characterize the OOD that can be reliably detected by our method.

- In Chapter 3, we first explore the practical challenges of training a differentially private flow generative model using DP-SGD on image datasets. Then we propose an efficient solution, i.e. DP-LFlow, by reducing the training from the full pixel space to a lower-dimensional latent space. The proposed method indicates robustness across different privacy budgets and different datasets. Uniquely, DP-LFlow is the first

related work to present high-resolution (up to 256×256) image synthesis with DP constraints.

Due to the inapplicability of batch normalization in DP-SGD, we currently remove the batch normalization layers in flow models. We also discussed applying group normalization as the alternative normalization layers. In the future, we may consider designing a tailored normalization layer that is suitable for running DP-SGD, which may not only benefit training a differentially private flow generative model but also other differentially private model that contains batch normalization. Additionally, improving the utility of large models when using DP-SGD is a potential area for further exploration.

- In Chapter 4, we explore the kernel method for training a DPGM by making the loss function in the RKHS differentially private, offering better scalability than directly applying DP-SGD to a generative model. To enhance the utility of existing kernel-based methods that truncate RKHS into a finite-dimensional space and apply vector RDP mechanisms, we generalize RDP to functional output and propose a novel improvement by applying functional Gaussian mechanisms to the loss function without truncating the RKHS. Our method consistently outperforms other SOTA kernel- and non-kernel-based methods by a large margin, especially when epsilon is small.

  Currently, we implement our method using the Gaussian kernel (which is the most commonly used kernel) and the associated RKHS. In the future, we may consider discussing the differences among different kernels and RKHS. Besides, the proposed method is validated on a few small-scale datasets, so some elementary generative neural networks are sufficient to synthesize decent samples. We believe the generation quality will be further improved if we utilize more advanced vision generative architecture, e.g. vision transformer, as our method is theoretically scalable to larger model and larger-scale datasets.

References

Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang (2016). "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318.

Abati, D., A. Porrello, S. Calderara, and R. Cucchiara (2019). "Latent space autoregression for novelty detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 481–490.

Acs, G., L. Melis, C. Castelluccia, and E. De Cristofaro (2018). "Differentially private mixture of generative neural networks". *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1109–1121.

Andrews, J., T. Tanay, E. J. Morton, and L. D. Griffin (2016). "Transfer representation-learning for anomaly detection". In: *International Conference on Machine Learning*.

Augenstein, S., H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, and B. A. y Arcas (2020). "Generative Models for Effective ML on Private, Decentralized Datasets". In: *International Conference on Learning Representations*.

Azmandian, F., A. Yilmazer, J. G. Dy, J. A. Aslam, and D. R. Kaeli (2012). "GPU-accelerated feature selection for outlier detection using the local kernel density ratio". In: *IEEE 12th International Conference on Data Mining*, pp. 51–60.

Bagdasaryan, E., O. Poursaeed, and V. Shmatikov (2019). "Differential privacy has disparate impact on model accuracy". In: *Advances in neural information processing systems*.

Balog, M., I. Tolstikhin, and B. Schölkopf (2018). "Differentially Private Database Release via Kernel Mean Embeddings". In: *Proceedings of the 36th International Conference on Machine Learning*.

Bennett, J., S. Lanning, et al. (2007). "The Netflix prize". In: *Proceedings of KDD cup and workshop*, p. 35.

Bernau, D., J. Robl, and F. Kerschbaum (2022). "Assessing Differentially Private Variational Autoencoders under Membership Inference". arXiv:2204.07877.

Bishop, C. M. (1994). "Novelty detection and neural network validation". *IEE Proceedings-Vision, Image and Signal processing*, vol. 141, no. 4, pp. 217–222.

Bonneel, N., J. Rabin, G. Peyré, and H. Pfister (2015). "Sliced and Radon Wasserstein Barycenters of Measures". *Journal of Mathematical Imaging and Vision*, vol. 51, pp. 22–45.

Cao, T., A. Bie, A. Vahdat, S. Fidler, and K. Kreis (2021). "Don't Generate Me: Training Differentially Private Generative Models with Sinkhorn Divergence". In: *Advances in Neural Information Processing Systems 21*.

Chalapathy, R. and S. Chawla (2019). "Deep learning for anomaly detection: A survey". arXiv:1901.03407.

Chalapathy, R., E. Toth, and S. Chawla (2018). "Group anomaly detection using deep generative models". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 173–189.

Charpentier, A., R. Elie, and C. Remlinger (2021). "Reinforcement learning in economics and finance". *Computational Economics*, pp. 1–38.

Chen, D., T. Orekondy, and M. Fritz (2020). "GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators". In: *Advances in Neural Information Processing Systems*, pp. 12673–12684.

Chen, J., C. Lu, B. Chenli, J. Zhu, and T. Tian (2020). "Vflow: More expressive generative flows with variational data augmentation". In: *International Conference on Machine Learning*, pp. 1660–1669.

Chen, Q., C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaarfar, and H. Zhu (2018). "Differentially Private Data Generative Models". arxiv:1812.0227.

Chen, T., S. Kornblith, M. Norouzi, and G. Hinton (2020). "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*, pp. 1597–1607.

Chen, X. and S. Yu (2016). "A collaborative intrusion detection system against DDoS for SDN". *IEICE Transactions on Information and Systems*, vol. 99, no. 9, pp. 2395–2399.

Choi, H., E. Jang, and A. A. Alemi (2018). "Waic, but why? generative ensembles for robust anomaly detection". arXiv:1810.01392.

Clanuwat, T., M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha (2018). "Deep Learning for Classical Japanese Literature". In: *NeurIPS Workshop on Machine Learning for Creativity and Design*.

Csiszár, I. (1963). "Eine informationstheoretische Ungleichung und ihre Anwendung auf den Beweis der Ergodizität von Markoffschen Ketten". *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének közleményei*, vol. 8, pp. 85–108.

Cuesta-Albertos, J. A., R. Fraiman, and T. Ransford (2007). "A sharp form of the Cramer–Wold theorem". *Journal of Theoretical Probability*, vol. 20, no. 2, pp. 201–209.

Darling, D. A. (1957). "The Kolmogorov-Smirnov, Cramér-von Mises Tests". *The Annals of Mathematical Statistics*, vol. 28, no. 4, pp. 823–838.

De, S., L. Berrada, J. Hayes, S. L. Smith, and B. Balle (2022). "Unlocking high-accuracy differentially private image classification through scale". arXiv:2204.13650.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database". In: *IEEE conference on computer vision and pattern recognition*, pp. 248–255.

Diaz, M., H. Wang, F. P. Calmon, and L. Sankar (2019). "On the robustness of information-theoretic privacy measures and mechanisms". *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 1949–1978.

Dinh, L., J. Sohl-Dickstein, and S. Bengio (2017). "Density estimation using Real NVP". In: *International Conference on Learning Representations*.

Dockhorn, T., T. Cao, A. Vahdat, and K. Kreis (2023). "Differentially private diffusion models". *Transactions on Machine Learning Research*.

Dong, J., A. Roth, and W. J. Su (2019). "Gaussian differential privacy". arXiv:1905.02383.

Dwork, C. (2006). "Differential privacy". In: *International Colloquium on Automata, Languages, and Programming*, pp. 1–12.

Dwork, C., A. Roth, et al. (2014). "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407.

Emmott, A., S. Das, T. Dietterich, A. Fern, and W.-K. Wong (2015). "A meta-analysis of the anomaly detection problem". *arXiv:1503.01158*.

Erfani, S. M., S. Rajasegarar, S. Karunasekera, and C. Leckie (2016). "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning". *Pattern Recognition*, vol. 58, pp. 121–134.

Esipova, M. S., A. A. Ghomi, Y. Luo, and J. C. Cresswell (2023). "Disparate Impact in Differential Privacy from Gradient Misalignment". In: *The Eleventh International Conference on Learning Representations*.

Fredrikson, M., S. Jha, and T. Ristenpart (2015). "Model inversion attacks that exploit confidence information and basic countermeasures". In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333.

Friedman, J. H., W. Stuetzle, and A. Schroeder (1984). "Projection Pursuit Density Estimation". *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 599–608.

Ghalebikesabi, S. et al. (2023). "Differentially private diffusion models generate useful synthetic images". arXiv:2302.13861.

Givens, G. H. and J. A. Hoeting (2012). "Computational statistics". Vol. 703.

Goldberger, A. L. et al. (2000). "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals". *circulation*, vol. 101, no. 23, e215–e220.

Goldwasser, S. and S. Micali (1984). "Probabilistic encryption". *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*. Vol. 27.

Graham, M. S., W. H. Pinaya, P.-D. Tudosiu, P. Nachev, S. Ourselin, and J. Cardoso (2023). "Denoising diffusion models for out-of-distribution detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2948–2957.

Grcić, M., I. Grubišić, and S. Šegvić (2021). "Densely connected normalizing flows". *arXiv:2106.04627*.

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola (2012). "A kernel two-sample test". *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773.

Hall, R., A. Rinaldo, and L. Wasserman (2013). "Differential privacy for functions and functional data". *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 703–727.

Harder, F., K. Adamczewski, and M. Park (2021). "DP-MERF: Differentially Private Mean Embeddings with Random Features for Practical Privacy-preserving Data Generation". In: *International Conference on Artificial Intelligence and Statistics*, pp. 1819–1827.

Havtorn, J. D. D., J. Frellsen, S. Hauberg, and L. Maaløe (2021). "Hierarchical VAEs Know What They Don't Know". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139, pp. 4117–4128.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Herbei, R. and M. H. Wegkamp (2006). "Classification with reject option". *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pp. 709–721.

Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2017). "GANs trained by a two time-scale update rule converge to a local Nash equilibrium". In: *Advances in Neural Information Processing Systems*. Vol. 30.

Ichimaru, Y. and G. Moody (1999). "Development of the polysomnographic database on CD-ROM". *Psychiatry and clinical neurosciences*, vol. 53, no. 2, pp. 175–177.

Ionescu, R. T., F. S. Khan, M.-I. Georgescu, and L. Shao (2019). "Object-centric auto-encoders and dummy anomalies for abnormal event detection in video". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7842–7851.

Jaini, P., I. Kobyzev, Y. Yu, and M. Brubaker (2020). "Tails of Lipschitz Triangular Flows". In: *International Conference on Machine Learning*.

Jiang, D. and S. Sun (2023). "DP-LFlow: Differentially private latent flow for scalable sensitive image generation". *Transactions on Machine Learning Research*.

Jiang, D., S. Sun, and Y. Yu (2022). "Revisiting flow generative models for out-of-distribution detection". In: *International Conference on Learning Representations*.

— (2023). "Functional Renyi Differential Privacy for Generative Modeling". *Advances in Neural Information Processing Systems*, vol. 36.

Jiang, D., G. Zhang, M. Karami, X. Chen, Y. Shao, and Y. Yu (2022). "DP^2-VAE: Differentially Private Pre-trained Variational Autoencoders". arXiv:2208.03409.

Jordon, J., J. Yoon, and M. Van Der Schaar (2019). "PATE-GAN: Generating synthetic data with differential privacy guarantees". In: *International conference on learning representations*.

Justel, A., D. Peña, and R. Zamar (1997). "A multivariate Kolmogorov-Smirnov test of goodness of fit". *Statistics & Probability Letters*, vol. 35, no. 3, pp. 251–259.

Kaplan, J. et al. (2020). "Scaling laws for neural language models". arXiv:2001.08361.

Karras, T., T. Aila, S. Laine, and J. Lehtinen (2018). "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*.

Kingma, D. P. and P. Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Vol. 31.

Kingma, D. P. and M. Welling (2013). "Auto-encoding variational Bayes". arXiv:1312.6114.

Kirichenko, P., P. Izmailov, and A. G. Wilson (2020). "Why Normalizing Flows Fail to Detect Out-of-Distribution Data". In: *Advances in Neural Information Processing Systems*. Vol. 33.

Kolouri, S., Y. Zou, and G. K. Rohde (2016). "Sliced Wasserstein Kernels for Probability Distributions". In: *CVPR*.

Krizhevsky, A., G. Hinton, et al. (2009). "Learning multiple layers of features from tiny images".

Lake, B. M., R. Salakhutdinov, and J. B. Tenenbaum (2015). "Human-level concept learning through probabilistic program induction". *Science*, vol. 350, no. 6266, pp. 1332–1338.

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

Lee, J., M. Kim, Y. Jeong, and Y. Ro (2022). "Differentially Private Normalizing Flows for Synthetic Tabular Data Generation". In: *AAAI Conference on Artificial Intelligence*.

Lewis, D. D., Y. Yang, T. Russell-Rose, and F. Li (2004). "Rcv1: A new benchmark collection for text categorization research". *Journal of machine learning research*, vol. 5, pp. 361–397.

Li, C.-L., W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos (2017). "Mmd gan: Towards deeper understanding of moment matching network". In: *Proceedings of the 30th Advances in neural information processing systems*.

Li, N., T. Li, and S. Venkatasubramanian (2007). "t-closeness: Privacy beyond k-anonymity and l-diversity". In: *IEEE 23rd International Conference on Data Engineering*, pp. 106–115.

Li, N., M. Lyu, D. Su, and W. Yang (2016). "Differential privacy: From theory to practice". *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 4, pp. 1–138.

Li, Y., K. Swersky, and R. Zemel (2015). "Generative moment matching networks". In: *International conference on machine learning*, pp. 1718–1727.

Liese, F. and I. Vajda (1987). "Convex Statistical Distances". Teubner, Leipzig.

Liew, S. P., T. Takahashi, and M. Ueno (2022). "PEARL: Data Synthesis via Private Embeddings and Adversarial Reconstruction Learning". In: *International Conference on Learning Representations*.

Liu, Z., J. P. Zhou, Y. Wang, and K. Q. Weinberger (2023). "Unsupervised out-of-distribution detection with diffusion inpainting". In: *International Conference on Machine Learning*, pp. 22528–22538.

Liu, Z., P. Luo, X. Wang, and X. Tang (2015). "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*.

Liutkus, A., U. Simsekli, S. Majewski, A. Durmus, and F.-R. Stöter (2019). "Sliced-Wasserstein Flows: Nonparametric Generative Modeling via Optimal Transport and Diffusions". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 4104–4113.

Long, Y., B. Wang, Z. Yang, B. Kailkhura, A. Zhang, C. A. Gunter, and B. Li (2021). "G-PATE: Scalable Differentially Private Data Generator via Private Aggregation of Teacher Discriminators". In: *Advances in Neural Information Processing Systems*.

Luo, Z., D. J. Wu, E. Adeli, and L. Fei-Fei (2021). "Scalable differential privacy with sparse network finetuning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5059–5068.

Machanavajjhala, A., D. Kifer, J. Gehrke, and M. Venkitasubramaniam (2007). "l-diversity: Privacy beyond k-anonymity". *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 3–es.

Markou, M. and S. Singh (2003a). "Novelty detection: a review—part 1: statistical approaches". *Signal processing*, vol. 83, no. 12, pp. 2481–2497.

— (2003b). "Novelty detection: a review—part 2:: neural network based approaches". *Signal processing*, vol. 83, no. 12, pp. 2499–2521.

McSherry, F. D. (2009). "Privacy integrated queries: an extensible platform for privacy-preserving data analysis". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30.

Mironov, I. (2017). "Rényi differential privacy". In: *IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275.

Mironov, I., K. Talwar, and L. Zhang (2019). "Rényi differential privacy of the sampled gaussian mechanism". arXiv:1908.10530.

Morningstar, W., C. Ham, A. Gallagher, B. Lakshminarayanan, A. Alemi, and J. Dillon (2021). "Density of States Estimation for Out of Distribution Detection". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130, pp. 3232–3240.

Muandet, K. and B. Schölkopf (2013). "One-class Support Measure Machines for Group Anomaly Detection". In: *29th Conference on Uncertainty in Artificial Intelligence*, pp. 449–458.

Nalisnick, E., A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan (2019). "Do Deep Generative Models Know What They Don't Know?" In: *International Conference on Learning Representations*.

Nalisnick, E., A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan (2019). "Detecting out-of-distribution inputs to deep generative models using typicality". In: *NeurIPS workshop on Bayesian Deep Learning*.

Narayanan, A. and V. Shmatikov (2008). "Robust de-anonymization of large sparse datasets". In: *IEEE Symposium on Security and Privacy*, pp. 111–125.

Nassif, A. B., I. Shahin, I. Attili, M. Azzeh, and K. Shaalan (2019). "Speech recognition using deep neural networks: A systematic review". *IEEE access*, vol. 7, pp. 19143–19165.

Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Nguyen, K., N. Ho, T. Pham, and H. Bui (2021). "Distributional Sliced-Wasserstein and Applications to Generative Modeling". In: *International Conference on Learning Representations*.

Nitsch, J., M. Itkina, R. Senanayake, J. Nieto, M. Schmidt, R. Siegwart, M. J. Kochenderfer, and C. Cadena (2021). "Out-of-distribution detection for automotive perception". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2938–2943.

Pang, G., L. Cao, L. Chen, and H. Liu (2017). "Learning Homophily Couplings from Non-IID Data for Joint Feature Selection and Noise-Resilient Outlier Detection." In: *International Joint Conference on Artificial Intelligence*, pp. 2585–2591.

Pang, G., L. Cao, L. Chen, and H. Liu (2018). "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2041–2050.

Pang, G., C. Shen, L. Cao, and A. V. D. Hengel (2021). "Deep learning for anomaly detection: A review". *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38.

Pang, G., C. Yan, C. Shen, A. v. d. Hengel, and X. Bai (2020). "Self-Trained Deep Ordinal Regression for End-to-End Video Anomaly Detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Papernot, N., M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar (2017). "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data". In: *International conference on learning representations*.

Papernot, N., S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson (2018). "Scalable Private Learning with PATE". In: *International Conference on Learning Representations*.

Paty, F.-P. and M. Cuturi (2019). "Subspace Robust Wasserstein Distances". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 5072–5081.

Pevnỳ, T. (2016). "Loda: Lightweight on-line detector of anomalies". *Machine Learning*, vol. 102, no. 2, pp. 275–304.

Pfitzner, B. and B. Arnrich (2022). "DPD-fVAE: Synthetic Data Generation Using Federated Variational Autoencoders With Differentially-Private Decoder". arXiv:2211.11591.

Pimentel, M. A., D. A. Clifton, L. Clifton, and L. Tarassenko (2014). "A review of novelty detection". *Signal processing*, vol. 99, pp. 215–249.

Rahimi, A. and B. Recht (2007). "Random features for large-scale kernel machines". In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 1177–1184.

Ren, J., P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan (2019). "Likelihood ratios for out-of-distribution detection". In: *Advances in Neural Information Processing Systems*.

Rényi, A. (1961). "On measures of entropy and information". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1, pp. 547–562.

Rezende, D. and S. Mohamed (2015). "Variational inference with normalizing flows". In: *International conference on machine learning*, pp. 1530–1538.

Rombach, R., A. Blattmann, D. Lorenz, P. Esser, and B. Ommer (2022). "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695.

Schirrmeister, R., Y. Zhou, T. Ball, and D. Zhang (2020). "Understanding Anomaly Detection with Deep Invertible Networks through Hierarchies of Distributions and Features". In: *Advances in Neural Information Processing Systems*. Vol. 33.

Serrà, J., D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque (2020). "Input complexity and out-of-distribution detection with likelihood-based generative models". In: *International Conference on Learning Representations*.

Shorten, C. and T. M. Khoshgoftaar (2019). "A survey on image data augmentation for deep learning". *Journal of Big Data*, vol. 6, no. 1, pp. 1–48.

Sohl-Dickstein, J., E. Weiss, N. Maheswaranathan, and S. Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*, pp. 2256–2265.

Song, J., Y. Song, and S. Ermon (2019). "Unsupervised out-of-distribution detection with batch normalization". arXiv:1910.09115.

Sriperumbudur, B. K., K. Fukumizu, and G. R. Lanckriet (2011). "Universality, Characteristic Kernels and RKHS Embedding of Measures". *Journal of Machine Learning Research*, vol. 12, no. 7.

Stein, G. et al. (2023). "Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models". arXiv:2306.04675.

Subramani, P., N. Vadivelu, and G. Kamath (2021). "Enabling Fast Differentially Private SGD via Just-in-Time Compilation and Vectorization". In: *NeurIPS*.

Sweeney, L. (2002). "k-anonymity: A model for protecting privacy". *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Szummer, M. and T. Jaakkola (2002). "Information regularization with partially labeled data". *Advances in Neural Information processing systems*, vol. 15.

Tabak, E. G. and E. Vanden-Eijnden (2010). "Density estimation by dual ascent of the log-likelihood". *Communications in Mathematical Sciences*, vol. 8, no. 1, pp. 217–233.

Takahashi, T., S. Takagi, H. Ono, and T. Komatsu (2020). "Differentially Private Variational Autoencoders with Term-wise Gradient Aggregation". arXiv:2006.11204.

Torkzadehmahani, R., P. Kairouz, and B. Paten (2019). "DP-CGAN: Differentially Private Synthetic Data and Label Generation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 98–104.

Toth, E. and S. Chawla (2018). "Group deviation detection methods: a survey". *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–38.

Tudor Ionescu, R., S. Smeureanu, B. Alexe, and M. Popescu (2017). "Unmasking the abnormal events in video". In: *IEEE International Conference on Computer Vision*, pp. 2895–2903.

Van den Oord, A., N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. (2016). "Conditional image generation with pixelcnn decoders". *Advances in neural information processing systems*, vol. 29.

Van Erven, T. and P. Harremos (2014). "Rényi divergence and Kullback-Leibler divergence". *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820.

Varshney, K. R. (2022). "Trustworthy Machine Learning". *Chappaqua, NY, USA: Independently Published*.

Vinaroz, M., M.-A. Charusaie, F. Harder, K. Adamczewski, and M. J. Park (2022). "Hermite Polynomial Features for Private Data Generation". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162, pp. 22300–22324.

Voulodimos, A., N. Doulamis, A. Doulamis, and E. Protopapadakis (2018). "Deep learning for computer vision: A brief review". *Computational intelligence and neuroscience*.

Waites, C. and R. Cummings (2021). "Differentially Private Normalizing Flows for Privacy-Preserving Density Estimation". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 1000–1009.

Wang, B. and N. Hegde (2019). "Privacy-preserving q-learning with functional noise in continuous spaces". In: *Proceedings of the 32nd Advances in Neural Information Processing Systems*.

Wang, B., F. Wu, Y. Long, L. Rimanic, C. Zhang, and B. Li (2021). "Datalens: Scalable privacy preserving training via gradient compression and aggregation". In: *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 2146–2168.

Wang, J., S. Sun, and Y. Yu (2019). "Multivariate triangular quantile maps for novelty detection". In: *Advances in Neural Information Processing Systems*. Vol. 32.

Weggenmann, B., V. Rublack, M. Andrejczuk, J. Mattern, and F. Kerschbaum (2022). "DP-VAE: Human-Readable Text Anonymization for Online Reviews with Differentially Private Variational Autoencoders". In: *Proceedings of the ACM Web Conference 2022*, pp. 721–731.

Xiao, H., K. Rasul, and R. Vollgraf (2017). "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms". arXiv:1708.07747.

Xie, L., K. Lin, S. Wang, F. Wang, and J. Zhou (2018). "Differentially Private Generative Adversarial Network". arxiv:1802.06739.

Xiong, L., B. Poczos, J. Schneider, A. Connolly, and J. VanderPlas (2011). "Hierarchical Probabilistic Models for Group Anomaly Detection". In: *International Conference on Artificial Intelligence and Statistics*. Vol. 15, pp. 789–797.

Xu, D., E. Ricci, Y. Yan, J. Song, and N. Sebe (2015). "Learning deep representations of appearance and motion for anomalous event detection". In: *Proceedings of British Machine Vision Conference*.

Xu, D., W. Du, and X. Wu (2021). "Removing disparate impact on model accuracy in differentially private stochastic gradient descent". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1924–1932.

Yu, D., H. Zhang, W. Chen, and T.-Y. Liu (2021). "Do Not Let Privacy Overbill Utility: Gradient Embedding Perturbation for Private Learning". In: *International Conference on Learning Representations*.

Yu, F., A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao (2015). "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop". arXiv:1506.03365.

Zhang, L., M. Goldstein, and R. Ranganath (2021). "Understanding Failures in Out-of-Distribution Detection with Deep Generative Models". In: *International Conference on Machine Learning*, pp. 12427–12436.

Zhang, Y., W. Liu, Z. Chen, J. Wang, Z. Liu, K. Li, and H. Wei (2020). "Out-of-distribution detection with distance guarantee in deep generative models". arXiv:2002.03328v3.

Zhou, J. T., J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh (2019). "AnomalyNet: An anomaly detection network for video surveillance". *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2537–2550.

APPENDICES

Appendix A

Appendix of Chapter 2

A.1 Datasets

In this section, we briefly introduce the public datasets we use and our preprocessing in this work. Note that image size is represented in $\#channels \times height \times width$, and all images are normalized into $[0.0, 1.0]$.

- Grayscale image datasets:
  - MNIST (LeCun et al. 1998): MNIST is a dataset of handwritten digits, including 10 classes (from digit 0 to 9) and 70000 images in total. Each image is in $1 \times 28 \times 28$.
  - FMNIST (Xiao et al. 2017): FMNIST is a dataset of Zalando's article images with 10 classes of clothes and shoes. There are 70000 images in total, and each of them is in $1 \times 28 \times 28$.
  - KMNIST (Clanuwat et al. 2018): KMNIST is a dataset of Japanese characters. There are 10 character labels and 70000 images in total, and each of them is in $1 \times 28 \times 28$.
  - Omniglot (Lake et al. 2015): Omniglot is a dataset of handwritten digits from different alphabets. There are 1623 different characters and 32460 images in total, and each of them is in $1 \times 105 \times 105$. We center-cropped images to the size of $1 \times 84 \times 84$, and then resize them into $1 \times 28 \times 28$.
- RGB image datasets:
  - CIFAR-10/100 (Krizhevsky, Hinton, et al. 2009): CIFAR-10/100 are datasets of natural images, e.g. animals and vehicles. CIFAR-10 contains 10 classes, while CIFAR-100 contains 100 classes. Both datasets contain a total of 60000 images. Each image is in $3 \times 32 \times 32$.

- SVHN (Netzer et al. 2011): SVHN is a dataset of street view housing numbers, including 10 classes and 99289 images in total. Each image is in $3 \times 32 \times 32$.
- LSUN (F. Yu et al. 2015): LSUN is a dataset of 10 scene categories, e.g., bedroom, classroom, kitchen. Only 10000 test images are used in this work. Images are not in an universal size. The smaller dimension is resized to 256 by the original dataset. We center-cropped images to the size of $3 \times 224 \times 224$, and then resize them into $3 \times 32 \times 32$.
- CelebA (Liu, P. Luo, et al. 2015): CelebA is a dataset of face images of celebrities. It contains a total of 10177 identities and 202599 face images. Each image is in $3 \times 178 \times 218$. We center-cropped images to the size of $3 \times 178 \times 178$, and then resize them into $3 \times 32 \times 32$. 50000 images are randomly sampled from predefined training split as the trainig set in this work.

We use the official training and test split for all datasets, and we create the validation set by randomly holding out 10% from the training split.

A.2 Implementation Details on Our Method

- Glow: Our Pytorch implementation of Glow was derived from Joost van Amersfoort's repository[1]. In our main results, for grayscale image datasets, we use 2 blocks, where each block consists of 3 steps of flow with 64 hidden channels. For RGB image datsets, we use 3 blocks, where each block consists of 3 steps of flow with 64 hidden channels. Affine coupling layers with squeezing factor of 2 are used for all datasets. All priors are set as standard Gaussians. The learning rate is set as 1e-5 for grayscale image datasets and 1e-3 for RGB image datasets. The optimizer is Adam with a weight decay of 1e-6.
- RealNVP: Our Pytorch implementation of RealNVP was derived from Ilya Kostrikov's repository[2]. In our main results, for grayscale image datasets, we use one block with 512 hidden channels. For RGB image datsets, we use 1 block with 2048 hidden channels. All priors are set as standard Gaussians. The learning rate is set as 1e-6 for grayscale image datasets and 1e-5 for RGB image datasets. The optimizer is Adam with a weight decay of 1e-6.

As mentioned earlier, the training epochs can significantly affect the OOD detection performance, and we found that the two SOTA baselines, as well as our GOOD1KS, are more sensitive to training time than GOOD2KS. We use validation set to select the training epoch that optimizes our GOOD2KS performance. Specifically, for RealNVP, we found that training the model for 50 epochs on FMNIST, CIFAR-100, CelebA, and 100 epochs

[1] https://github.com/y0ast/Glow-PyTorch
[2] https://github.com/ikostrikov/pytorch-flows

Algorithm 3: Group OOD detection based on two-sample KS test (GOOD2KS).

Input: Training InD samples $\mathbf{X}_{\text{train}} \in \mathbb{R}^{d \times N}$ (N denotes the size of training set), Test OOD samples $\mathbf{X}_{\text{test}}$ splitted into m groups $\mathbf{X}_1, \cdots, \mathbf{X}_m$ with each $\mathbf{X}_i \in \mathbb{R}^{d \times b}$ (b for batch size and d for dimension); random projection matrix $\mathbf{W} \in \mathbb{R}^{d \times n}$.

$\mathbf{Z}_{\text{train}} = \mathsf{T}^{-1}(\mathbf{X}_{\text{train}}) \in \mathbb{R}^{d \times N}$

$\mathbf{S}_{\text{train}} = \mathbf{W}^\top \mathbf{Z}_{\text{train}} \in \mathbb{R}^{n \times N}$

for $i \leftarrow 1$ ***to*** m **do**

  $\mathbf{Z}_i = \mathsf{T}^{-1}(\mathbf{X}_i) \in \mathbb{R}^{d \times b}$ `// transform into the latent space`

  $\mathbf{S}^{(i)} = \mathbf{W}^\top \mathbf{Z}_i \in \mathbb{R}^{n \times b}$ `// project onto` n `random directions`

  for $j \leftarrow 1$ ***to*** n **do**

    $k_{ij} = \text{KS}(\mathbf{S}^{(i)}_{j:}, \mathbf{S}_{\text{train}(j:)})$ `// conduct two-sample KS test`

  $k_i \leftarrow \frac{1}{n} \sum_{j=1}^{n} k_{ij}$ `// average over` n `random directions`

compute AUROC for $\mathbf{X}_{\text{test}}$ based on all k_i's

on CIFAR-10, SVHN, achieves better GOOD2KS results. For Glow, we found that training the model for 2 epochs on CIFAR-10, 5 epochs on CIFAR-100, and 200 epochs on FMNIST, SVHN, CelebA, achieves better GOOD2KS results.

We also present the detailed algorithm of our GOOD2KS for completeness, as shown in Algorithm 3.

A.3 Implementation Details of Benchmarks

For TyTest, we follow their Algorithm 1 except the threshold ϵ to calculate the detection score, since we are using AUROC for comparison. For KLOD, we follow their equation (4) to calculate the detection score. However, we observe one empirical issue of KLOD, i.e. the estimation of a high-dimensional covariance matrix from a small batch are mostly singular, which leads the log determinant of covariance matrix in equation (4) of Y. Zhang et al. 2020 to be negative infinity, and thus not indicative as a detection score at all. In our experiment, we calculate $\mu^\top \mu$ to replicate their results.

A.4 Framework

Our framework is illustrated in Figure A.1. When the flow model T^{-1} is trained, we do the inference to obtain the latent variable, i.e. $z = \mathsf{T}^{-1}(x)$, then do the detection based on the distribution of z.

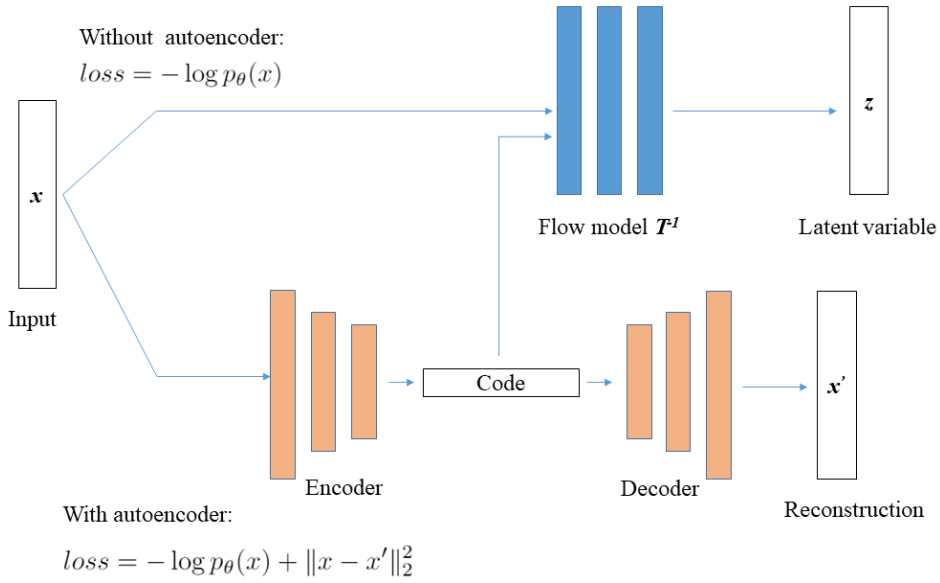


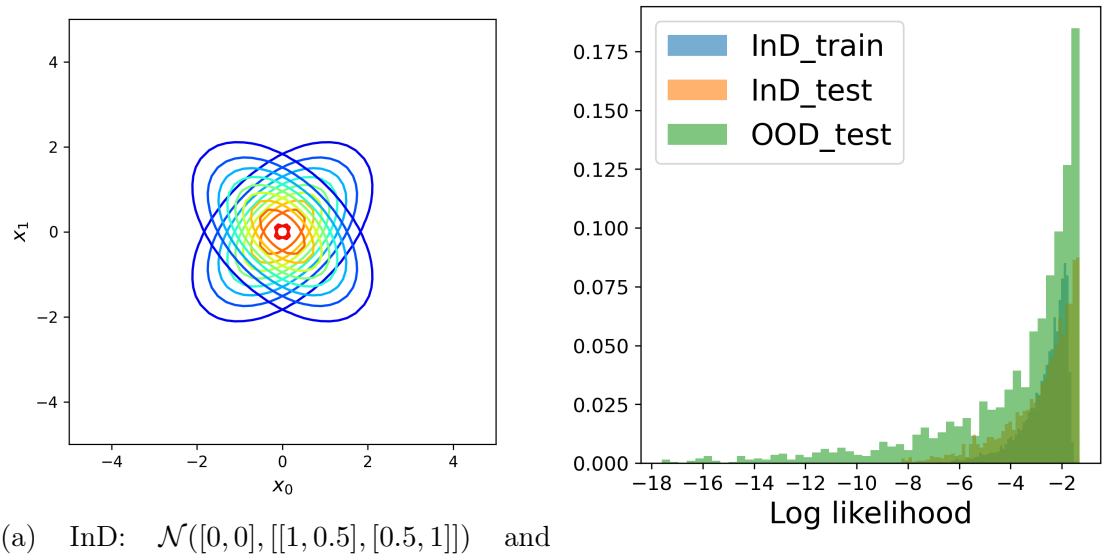
Figure A.1: The framework of our method.

A.5 Motivation for Groupwise Detection

As mentioned in section 2.3.1, distribution based group anomaly is only detectable by groupwise detection methods. We conduct a synthetic experiment to illustrate this point. Consider two 2-d Gaussians with the same zero mean and orthogonal eigenvectors, e.g. InD=$\mathcal{N}([0,0],[[1,0.5],[0.5,1]])$ and OOD=$\mathcal{N}([0,0],[[1,-0.5],[-0.5,1]])$. The two distributions have a large overlap around origin where the majority of mass reside in (they have the same density peak). Therefore, if we train a generative model (e.g. RealNVP in this example) to estimate their density, the InD and OOD density histograms will largely overlap, as shown in Figure A.2, which results in an AUROC of pointwise likelihood detection score as low as 0.57. In contrast, with increasing number of test samples drawn from these two distributions, the separation of InD and OOD will be increasingly clear (which is true for both sample space and latent space). Our GOOD1KS and GOOD2KS methods achieve detection AUROC of 0.67|0.67 for batch size = 10, 0.77|0.77 for batch size = 20, and 0.97|0.96 for batch size = 100, respectively.

A.6 Point OOD Detection via Data Augmentation

In GOOD, we assume that a group of test samples are either from the in-distribution or not. Now we will relax this assumption and extend the proposed group OOD detection to



(a) InD: $\mathcal{N}([0,0],[[1,0.5],[0.5,1]])$ and OOD: $\mathcal{N}([0,0],[[1,-0.5],[-0.5,1]])$

(b) Log likelihood

Figure A.2: An example of distribution based group anomaly. (a) Two 2-d Gaussians with similar density and large overlap in support. (b) Density histograms largely overlap, where likelihood score only achieves a detection AUROC of 0.57.

point OOD detection.

The key of the extension is to obtain a group of samples based on one single test sample. To do so, one option is to adopt data augmentation (Shorten and Khoshgoftaar 2019), which has been extensively used in deep learning when the number of the training samples is limited. Data augmentation is also a common practice to construct the positive pairs in contrastive learning (T. Chen et al. 2020). Concretely, for each image sample we can construct a group of samples by applying, e.g., rotations, random cropping and resizing, and color distortion. For comparison, we produce 10 and 40 samples respectively. Specifically, 10 augmented samples consist of 3 rotated images (90, 180, 270 degrees), 3 random cropped and resized images, and 3 color distorted (jitter) images, as well as the original one, while 40 augmented samples consist of 3 rotated images (90, 180, 270 degrees), 18 random cropped and resized images, and 18 color distorted (jitter) images, as well as the original one. Generally, our detection performance improves with the number of data augmentation.

We compare the performance of our point OOD algorithm with two SOTA benchmarks: $\log q(\mathbf{x})$ and input complexity $\mathcal{S}$ (Serrà et al. 2020), as shown in Table A.1. Both $\log q(\mathbf{x})$ and $\mathcal{S}$ encounter complete failure detection case, e.g. CIFAR-10/SVHN for $\log q(\mathbf{x})$ and FMNIST/KMNIST for $\mathcal{S}$, while our POOD is more robust to different dataset pairs.

Table A.1: AUROC on RealNVP for point OOD detection (higher is better). Our results are denoted by R1|R2, where R1 corresponds to GOOD1KS, and R2 corresponds to GOOD2KS.

InD	OOD	$\log q(\mathbf{x})$	$\mathcal{S}$	Ours-10	Ours-40
FMNIST	MNIST	0.80	0.43	0.74|0.75	0.79|0.81
	KMNIST	0.91	0.19	0.84|0.83	0.90|0.89
	Omniglot	1.00	1.00	1.00|1.00	1.00|1.00
CIFAR10	SVHN	0.16	0.80	0.85|0.81	0.85|0.82
	CelebA	0.65	0.51	0.59|0.59	0.57|0.57
	LSUN	0.60	0.48	0.45|0.47	0.46|0.48

A.7 Additional Results

A.7.1 GOOD Results using RealNVP - AUPR

The corresponding AUPRs for the experiments in Table 2.1 are shown in Table A.2.

A.7.2 GOOD Results using Glow

The optimal number of random projections can be dataset- and model-dependent. We run a similar experiment as section 2.4.2 with Glow to select the optimal number of projections. The selection process is similar to section 2.4.2, thus is omitted. Finally, we use 100 random projections when training Glow on grayscale image datasets, and use 200 random projections when training Glow on RGB image datasets. The detection result is summarized in Table A.3. Comparing Table A.3 with Table 2.1, we can notice that the detection performance can be largely affected by the network architecture. In particular, for KLOD and our GOOD1KS the AUROC for FMNIST (InD) and MNIST/KMNIST (OOD) can be close to 0, which indicates a complete failure of detection. The underlying reason is that it is always questionable whether a flow model is well-trained, which means there is always the possibility that $\mathsf{T}^{-1}_{\#\text{OOD}}$ is closer to the latent prior than $\mathsf{T}^{-1}_{\#\text{InD}}$, motivating us to propose the two-sample version to improve the robustness.

A.7.3 Comparison of Different Divergence Measures

We conduct a few more experiments to compare KS test and CvM test, as shown in Table A.4, where the KS test indicates more robustness especially when batch size is small.

Table A.2: AUPR on RealNVP (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest AUPR are in boldface, and failure cases (where AUPR is below 0.5) are underlined.

InD	OOD	batch size = 5			batch size = 10		
		TyTest	KLOD	Ours	TyTest	KLOD	Ours
FMNIST	MNIST	0.65	0.86	**0.99**|**0.99**	0.68	0.89	**1.00**|**1.00**
	KMNIST	0.95	0.95	**0.99**|**0.99**	0.96	0.96	**1.00**|**1.00**
	Omniglot	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
CIFAR-10	SVHN	**0.95**	0.68	**0.95**|0.93	0.98	0.72	**1.00**|0.99
	CelebA	0.78	**0.94**	0.86|0.86	0.87	**0.99**	0.94|0.94
	LSUN	0.60	**0.67**	0.60|0.61	0.70	**0.78**	0.66|0.70
CIFAR-100	SVHN	**0.96**	0.63	**0.96**|0.94	0.99	0.68	**1.00**|0.99
	CelebA	0.60	**0.89**	0.85|0.85	0.63	**0.98**	0.93|0.93
	LSUN	<u>0.49</u>	**0.57**	0.56|**0.57**	0.52	**0.66**	0.60|0.64
SVHN	CIFAR-10	**1.00**	0.98	0.62|0.70	**1.00**	0.98	0.80|0.89
	CIFAR-100	**1.00**	0.98	0.64|0.71	**1.00**	0.99	0.82|0.90
	CelebA	**1.00**	**1.00**	0.79|0.83	**1.00**	**1.00**	0.93|0.96
	LSUN	**1.00**	0.99	0.70|0.74	**1.00**	**1.00**	0.88|0.93
CelebA	CIFAR-10	0.97	**0.98**	0.86|0.89	**1.00**	**1.00**	0.97|0.98
	CIFAR-100	**0.98**	**0.98**	0.85|0.88	**1.00**	**1.00**	0.96|0.98
	SVHN	0.83	0.85	**0.98**|0.97	0.86	0.99	**1.00**|**1.00**
	LSUN	**0.99**	**0.99**	0.81|0.85	**1.00**	**1.00**	0.96|0.98

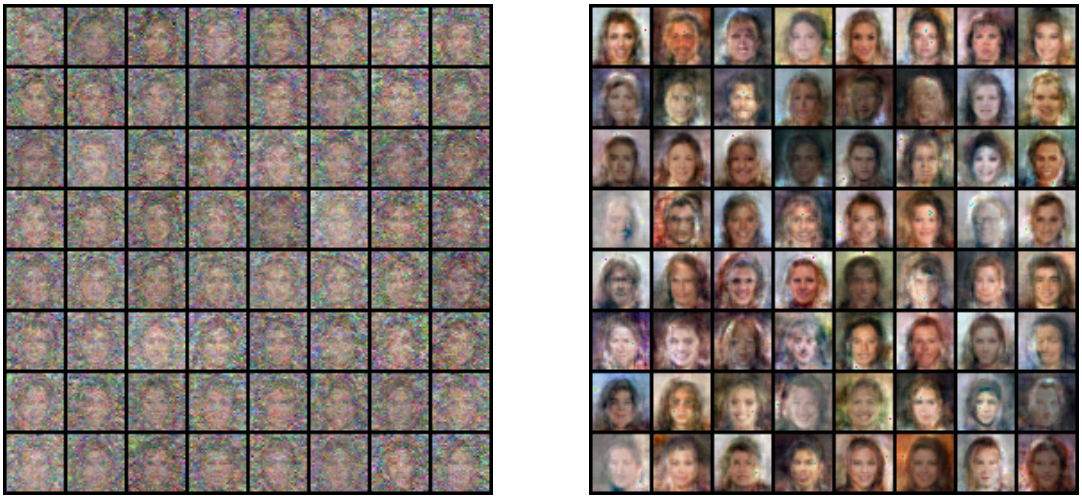
A.7.4 Batch Size = 20

We add more results when batch size is 20, as shown in Table A.6. The model is the same as Table 2.1 and A.3. As expected, the GOOD performance will increase as batch size increases. Our method can achieve an AUROC of 1.00 for most cases when batch size is 20.

A.7.5 Generation Quality vs. OOD Detection

Model capacity: We repeat the experiment regarding generation quality on RealNVP, and the comparison is shown in Figure A.3. Again, the batch size is fixed as 10. The observation is similar to Figure 2.7, i.e. a less capable model that generates more blurred images performs better in OOD detection.

Training time: While the OOD detection performance generally improves with training time, as shown in Figure A.4, we still found some anomalies. We found a few counterex-

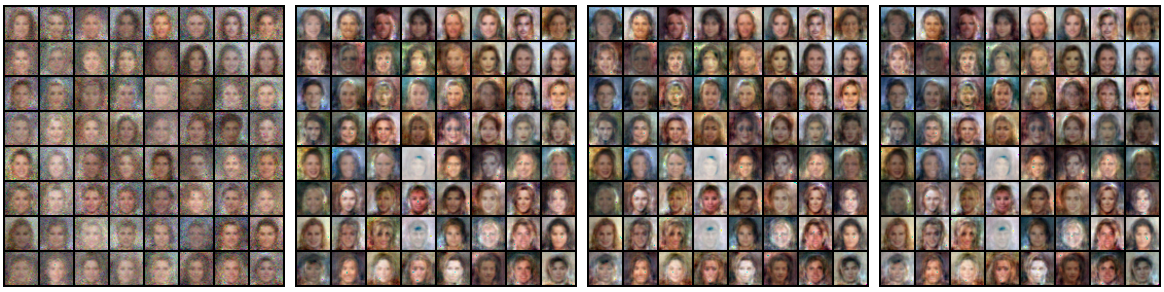


(a) (b)

Figure A.3: Model capacity vs. OOD detection. Images are generated from RealNVP trained on CelebA. Assume CIFAR-10 as OOD, and our GOOD1KS and GOOD2KS results are segmented by |. (a) Generated images from a RealNVP with 1 block and 2048 hidden channels. AUROCs = 0.98|0.99. (b) Generated images from a RealNVP with 16 blocks and 512 hidden channels. AUROCs = 0.88|0.88.

amples that the OOD detection performance fluctuate or even declines with training time, as shown in Figure A.5 and A.6.

This counter-intuitive phenomenon might be surprising at a first glance, but it is not completely baseless. Note that the OOD detection failure for FMNIST/MNIST and CIFAR-10/SVHN (the two most famous OOD detection failure cases as reported by Nalisnick, Matsukawa, Teh, Gorur, et al. (2019)) is because the generative model gives OOD input higher density estimation than InD, i.e. the log likelihood histogram of OOD sits on the right hand side of InD. A better model (either with a larger capacity or more training time) tends to give a more accurate estimation of density. However, the OOD detection is mainly based on both the separation of the InD score and OOD score and their relative order (i.e. InD must have higher likelihood estimation than OOD input, otherwise detection based on the likelihood will fail) instead of their respective accuracy. Some prior works (Schirrmeister et al. 2020; L. Zhang et al. 2021) reported a related result. Specifically, Schirrmeister et al. (2020) modifies the model architecture, so that the wrong relative order of InD and OOD density histograms is reversed, which leads to the success of OOD detection, although the modified model has worse log likelihood. In contrast, L. Zhang et al. (2021) consider a successful case, where the relative order of InD and OOD bpd (bits per dimension) histograms is correct, but they found that a mis-estimated model can lead to a smaller overlap in bpd histograms of InD and OOD, which results in the improvement of detection performance (see Figure 2 in L. Zhang et al. (2021)). This intuition is also valid in our method. Take our one-sample test as an example. For a worse model, the



(a) (b) (c) (d)

Figure A.4: Training epochs vs. OOD detection. Images are generated from RealNVP trained on CelebA. Assume CIFAR-10 as OOD, and our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 10 epochs. AUROCs = 0.78|0.78. (b) Trained for 100 epochs. AUROCs = 0.81|0.81. (c) Trained for 150 epochs. AUROCs = 0.86|0.86. (d) Trained for 200 epochs. AUROCs = 0.88|0.88.



(a) (b) (c) (d)

Figure A.5: Training epochs vs. OOD detection. Images are generated from a Glow with $K = 16, L = 3, h = 128$ trained on CIFAR-10. Treating SVHN as OOD, our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 5 epochs. AUROCs = 0.96|0.96. (b) Trained for 10 epochs. AUROCs = 0.96|0.95. (c) Trained for 100 epochs. AUROCs = 0.76|0.84. (d) Trained for 200 epochs. AUROCs = 0.87|0.84.

transformed distribution of both InD and OOD in the latent space will be further away from the prior, compared to a better model, but the separation in this case might be more pronounced, which is why a worse model can sometimes lead to better OOD detection.

More importantly, our detection method is not based on likelihood score, and yet we still observe a similar phenomenon. These results are indeed related, and our results corroborate existing work by examining more flow models, test statistics, real datasets and metrics.



(a) (b) (c) (d)

Figure A.6: Training epochs vs. OOD detection. Images are generated from a Glow with $K = 16, L = 2, h = 128$ trained on FMNIST. Treating MNIST as OOD, our GOOD1KS and GOOD2KS results are segmented by |. (a) Trained for 5 epochs. AUROCs = 1.00|1.00. (b) Trained for 10 epochs. AUROCs = 1.00|1.00. (c) Trained for 100 epochs. AUROCs = 0.90|1.00. (d) Trained for 200 epochs. AUROCs = 0.92|1.00.

A.7.6 Intra-dataset OOD Detection

Till now, all our OOD detection experiments were conducted across different datasets (i.e. inter-dataset OOD). We conduct additional experiments on detecting OOD within one dataset, i.e. treating one class as InD, the rest classes combined as OOD. Table A.7 shows the detection results on FMNIST and CIFAR-10. The batch size is fixed as 10, and our method uses 200 random projections. Compared to SOTA baselines, our method achieves the best intra-dataset OOD detection performance on FMNIST dataset for all classes, while on CIFAR-10 dataset, our method yields the near-best detection performance for all classes. Besides, there is no detection failure case in our method, which serves as another evidence of the effectiveness and generality of our proposed method.

A.7.7 OOD Detection on Time-series Data

Till now, all our experiments are conducted on widely used image benchmarks. It will be interesting to investigate whether our method can generalize beyond image datasets. We consider testing physiological signals (time-series data), e.g. Electrocardiography (ECG) and Electroencephalography (EEG). Clinically, ECG and EEG are electric signals that could reflect heart or brain activities, thus can be critical for the monitoring and diagnosis of cardiac dysfunctions or neural disorders. In this experiment, we train a RealNVP on one signal, and test the OOD detection performance by treating the other kind of signal as OOD. Our experimental data come from a publicly available database, i.e. MIT-BIH Polysomnographic database (Goldberger et al. 2000; Ichimaru and Moody 1999) [3]. The MIT-BIH Polysomnographic Database collects various physiological signals, including EEG

[3] https://physionet.org/content/slpdb/1.0.0/

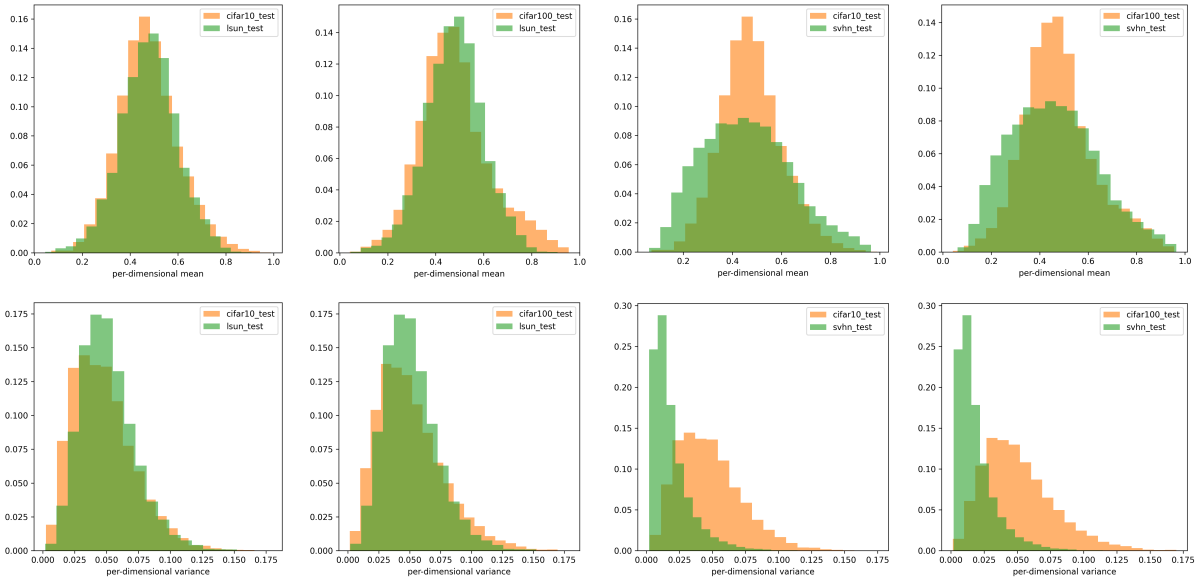
and ECG, from 16 subjects during sleep. We randomly select four records from the database for analysis, i.e. slp03, slp04, slp14, slp16, where each record is exactly 6-hour long. Amplitudes are normalized between 0 and 1. The whole signal is segmented into 10-second strips, so the size of this toy dataset is 8640, where 10% is randomly held out for validation, and another 10% is randomly held out for test. The rest 80% is used for training. The sampling rate is 250 Hz, so the data dimension is 2500. Table A.8 shows the detection performance of OOD detection on ECG/EEG and EEG/ECG by our method as well as two SOTA baselines, where the batch size is fixed as 10 and we use 200 random projections for our methods. It can be observed from Table A.8 that our method, especially GOOD2KS, indicates more robustness against SOTA baselines. In the EEG/ECG setting, typicality test fails, implying an overlap in density histograms of InD and OOD, and KLOD is slightly better than random guess. In contrast, our GOOD2KS achieves around 0.9 metrics in both experimental settings.

A.7.8 Investigation of CIFARs vs. LSUN

For CIFAR vs LSUN, our experimental results indicate that this is a challenging case for our method. We examined the raw distributions of the two datasets, through per-dimensional mean and variance. The detailed result is given in Figure A.7, where we verify that CIFARs and LSUN share a very close per-dimensional mean and variance, while for CIFARs vs. SVHN, although their per-dimensional mean share a similar range, their distributional shapes are different in terms of per-dimensional variances. This implies the CIFARs vs. LSUN pair is distributionally more similar and hence more challenging for our methods. The immediate solution is to increase the batch size. When batch size = 100, our methods (using RealNVP) can achieve AUROC of 0.97|1.00 for CIFAR-10 vs. LSUN, and 0.90|0.99 for CIFAR-100 vs. LSUN. When batch size is beyond 200, our methods can achieve perfect detection performance in both settings.

A.7.9 Comparison with Average Density

For thoroughness we also include comparison against a straightforward baseline, i.e. using the averaged density across a batch as the detection score (we denote this approach as Avg in Table A.5). We conduct additional experiments on FMNIST/MNIST, CIFAR-10/SVHN (the two most popular experimental settings in OOD detection) using both RealNVP and Glow, as shown in Table A.5. The batch size is fixed as 10. The results of other methods are directly cited from Table 2.1 and Table A.3. It can be observed that directly averaging the density across a batch is not competitive. The result is not surprising, because as long as the density histograms of InD and OOD sit in the wrong relative position, the average density will not work, regardless of batch size.



(a) CIFAR-10 vs. LSUN (b) CIFAR-100 vs. LSUN (c) CIFAR-10 vs. SVHN (d) CIFAR-100 vs. SVHN

Figure A.7: Per-dimensional mean and variance comparison. (a)-(b) Per-dimensional mean and variance of CIFARs and LSUN are close, which implies that CIFARs and LSUN share a similar distribution of normalized pixel values. (c)-(d) As a comparison, CIFARs vs SVHN have more clearly separable per-dimensional variance.

Table A.3: AUROC and AUPR on Glow (higher is better). Our results are denoted by GOOD1KS | GOOD2KS. Highest metrics are denoted in boldface, and failure cases (where metric is below 0.5) are underlined.

InD	OOD	Metric	batch size = 5			batch size = 10		
			TyTest	KLOD	Ours	TyTest	KLOD	Ours
FMNIST	MNIST	AUC	**0.99**	0.12	0.19|0.98	**1.00**	0.04	0.11|**1.00**
		AUPR	**0.98**	0.32	0.34|**0.98**	**1.00**	0.31	0.32|**1.00**
	KMNIST	AUC	0.51	0.38	0.01|**0.97**	0.55	0.13	0.00|**1.00**
		AUPR	0.53	0.41	0.31|**0.96**	0.56	0.33	0.30|**1.00**
	Omniglot	AUC	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
		AUPR	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
CIFAR-10	SVHN	AUC	**1.00**	0.96	0.98|0.96	1.00	1.00	1.00|1.00
		AUPR	**1.00**	0.98	0.99|0.98	1.00	1.00	1.00|1.00
	CelebA	AUC	0.55	**0.98**	0.71|0.86	0.68	**1.00**	0.79|0.96
		AUPR	0.69	**0.99**	0.82|0.92	0.77	**1.00**	0.88|0.98
	LSUN	AUC	**0.86**	0.82	0.31|0.60	**0.94**	0.80	0.18|0.58
		AUPR	**0.88**	0.81	0.38|0.58	**0.95**	0.78	0.33|0.56
CIFAR-100	SVHN	AUC	**1.00**	0.79	0.94|0.90	**1.00**	0.97	0.99|0.98
		AUPR	**1.00**	0.91	0.98|0.96	**1.00**	0.99	0.99|0.99
	CelebA	AUC	0.35	**0.97**	0.74|0.82	0.35	**1.00**	0.84|0.96
		AUPR	0.55	**0.98**	0.85|0.90	0.55	**1.00**	0.91|0.98
	LSUN	AUC	0.66	**0.82**	0.38|0.59	0.76	**0.86**	0.27|0.59
		AUPR	0.66	**0.80**	0.42|0.58	0.78	**0.85**	0.37|0.58
SVHN	CIFAR-10	AUC	**1.00**	**1.00**	0.10|0.92	**1.00**	**1.00**	0.01|0.99
		AUPR	**1.00**	**1.00**	0.16|0.84	**1.00**	**1.00**	0.15|0.98
	CIFAR-100	AUC	**1.00**	**1.00**	0.08|0.93	**1.00**	**1.00**	0.01|0.99
		AUPR	**1.00**	**1.00**	0.16|0.86	**1.00**	**1.00**	0.15|0.98
	CelebA	AUC	**1.00**	**1.00**	0.13|**1.00**	**1.00**	**1.00**	0.02|**1.00**
		AUPR	**1.00**	**1.00**	0.27|0.99	**1.00**	**1.00**	0.26|**1.00**
	LSUN	AUC	**1.00**	**1.00**	0.02|0.99	**1.00**	**1.00**	0.00|**1.00**
		AUPR	**1.00**	**1.00**	0.15|0.98	**1.00**	**1.00**	0.15|**1.00**
CelebA	CIFAR-10	AUC	0.26	0.68	0.36|**0.84**	0.14	0.67	0.27|**0.94**
		AUPR	0.23	0.53	0.27|**0.73**	0.20	0.52	0.23|**0.91**
	CIFAR-100	AUC	0.29	0.65	0.30|**0.82**	0.14	0.61	0.20|**0.94**
		AUPR	0.25	0.50	0.24|**0.71**	0.21	0.46	0.21|**0.90**
	SVHN	AUC	**1.00**	0.83	0.57|0.97	**1.00**	0.88	0.53|**1.00**
		AUPR	**1.00**	0.87	0.67|0.98	**1.00**	0.91	0.65|**1.00**
	LSUN	AUC	0.29	0.64	0.33|**0.85**	0.18	0.59	0.22|**0.96**
		AUPR	0.25	0.47	0.25|**0.75**	0.21	0.43	0.22|**0.92**

Table A.4: Comparison of KS test and CvM test on RealNVP. AUROC is shown (higher is better). Our results are denoted by GOOD1KS | GOOD2KS.

InD/OOD	batch size = 5		batch size = 10		batch size = 20	
	KS	CvM	KS	CvM	KS	CvM
CIFAR-10/SVHN	0.86|0.82	0.77|0.71	0.99|0.97	0.97|0.94	1.00|1.00	1.00|1.00
CIFAR-10/CelebA	0.80|0.80	0.82|0.82	0.92|0.93	0.94|0.95	0.99|0.99	1.00|1.00
CIFAR-10/LSUN	0.61|0.63	0.63|0.64	0.65|0.68	0.68|0.70	0.72|0.78	0.76|0.79
SVHN/CIFAR-10	0.89|0.93	0.88|0.90	0.95|0.98	0.93|0.96	0.99|1.00	0.97|0.99

Table A.5: Comparison with averaged density. Our results are denoted by GOOD1KS | GOOD2KS. We use the boldface for the highest metric, and underline the detection failure case (where metric is below 0.5).

InD/OOD	Metrics	RealNVP, batch size = 10				Glow, batch size = 10			
		Avg	TyTest	KLOD	Ours	Avg	TyTest	KLOD	Ours
FMNIST/MNIST	AUC	0.83	0.82	0.96	**1.00**|**1.00**	<u>0.00</u>	**1.00**	<u>0.04</u>	<u>0.11</u>|**1.00**
	AUPR	0.69	0.68	0.89	**1.00**|**1.00**	<u>0.31</u>	**1.00**	<u>0.31</u>	<u>0.32</u>|**1.00**
CIFAR-10/SVHN	AUC	<u>0.01</u>	0.95	<u>0.35</u>	**0.99**|0.98	<u>0.00</u>	**1.00**	**1.00**	**1.00**|**1.00**
	AUPR	0.51	0.98	0.72	**1.00**|0.99	0.51	**1.00**	**1.00**	**1.00**|**1.00**

Table A.6: AUROC and AUPR when batch size = 20. Our results are denoted by GOOD1KS | GOOD2KS. We use the boldface for the highest metrics, and underline the detection failure case (where metric is below 0.5).

InD	OOD	Metric	RealNVP, batch size = 20			Glow, batch size = 20		
			TyTest	KLOD	Ours	TyTest	KLOD	Ours
FMNIST	MNIST	AUC	0.83	0.97	**1.00**|**1.00**	**1.00**	0.01	0.05|**1.00**
		AUPR	0.67	0.93	**1.00**|**1.00**	0.99	0.31	0.31|**1.00**
	KMNIST	AUC	0.98	0.99	**1.00**|**1.00**	0.58	0.02	0.00|**1.00**
		AUPR	0.97	0.98	**1.00**|**1.00**	0.60	0.31	0.31|**1.00**
	Omniglot	AUC	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
		AUPR	1.00	1.00	1.00|1.00	1.00	1.00	1.00|1.00
CIFAR-10	SVHN	AUC	0.99	0.46	**1.00**|**1.00**	1.00	1.00	1.00|1.00
		AUPR	**1.00**	0.78	**1.00**|**1.00**	1.00	1.00	1.00|1.00
	CelebA	AUC	0.91	**1.00**	0.98|0.99	0.78	**1.00**	0.89|**1.00**
		AUPR	0.94	**1.00**	0.99|0.99	0.85	**1.00**	0.93|**1.00**
	LSUN	AUC	0.81	**0.91**	0.73|0.79	**0.99**	0.75	0.07|0.55
		AUPR	0.80	**0.90**	0.75|0.80	**0.99**	0.72	0.33|0.54
CIFAR-100	SVHN	AUC	**1.00**	0.41	**1.00**|**1.00**	1.00	1.00	1.00|1.00
		AUPR	**1.00**	0.75	**1.00**|**1.00**	1.00	1.00	1.00|1.00
	CelebA	AUC	0.53	**1.00**	0.98|0.98	0.32	**1.00**	0.94|**1.00**
		AUPR	0.66	**1.00**	0.99|0.99	0.54	**1.00**	0.97|**1.00**
	LSUN	AUC	0.60	**0.85**	0.69|0.76	0.87	**0.91**	0.17|0.63
		AUPR	0.58	**0.81**	0.69|0.76	0.88	**0.91**	0.34|0.62
SVHN	CIFAR-10	AUC	**1.00**	0.99	0.99|**1.00**	**1.00**	**1.00**	0.00|**1.00**
		AUPR	**1.00**	0.98	0.94|0.99	**1.00**	**1.00**	0.15|**1.00**
	CIFAR-100	AUC	**1.00**	**1.00**	0.99|**1.00**	**1.00**	**1.00**	0.00|**1.00**
		AUPR	**1.00**	0.99	0.95|0.99	**1.00**	**1.00**	0.15|**1.00**
	CelebA	AUC	1.00	1.00	1.00|1.00	**1.00**	**1.00**	0.00|**1.00**
		AUPR	1.00	1.00	1.00|1.00	**1.00**	**1.00**	0.26|**1.00**
	LSUN	AUC	1.00	1.00	1.00|1.00	**1.00**	**1.00**	0.00|**1.00**
		AUPR	**1.00**	**1.00**	0.99|**1.00**	**1.00**	**1.00**	0.16|**1.00**
CelebA	CIFAR-10	AUC	1.00	1.00	1.00|1.00	0.05	0.67	0.17|**1.00**
		AUPR	1.00	1.00	1.00|1.00	0.19	0.54	0.21|**0.99**
	CIFAR-100	AUC	1.00	1.00	1.00|1.00	0.05	0.57	0.10|**0.99**
		AUPR	1.00	1.00	1.00|1.00	0.19	0.42	0.20|**0.99**
	SVHN	AUC	0.85	**1.00**	**1.00**|**1.00**	**1.00**	0.93	0.50|**1.00**
		AUPR	0.89	**1.00**	**1.00**|**1.00**	**1.00**	0.95	0.62|**1.00**
	LSUN	AUC	1.00	1.00	1.00|1.00	0.10	0.58	0.12|**1.00**
		AUPR	1.00	1.00	1.00|1.00	0.20	0.44	0.20|**1.00**

Table A.7: Intra-dataset OOD detection using RealNVP. AUROC and AUPR are shown (higher is better). The batch size is fixed as 10. Our results are denoted by GOOD1KS|GOOD2KS.

Classes	Metrics	FMNIST			CIFAR-10		
		TyTest	KLOD	Ours	TyTest	KLOD	Ours
0	AUC	0.99	**1.00**	**1.00**|**1.00**	0.93	**0.98**	0.94|0.97
	AUPR	1.00	1.00	1.00|1.00	0.99	**1.00**	0.99|0.99
1	AUC	1.00	1.00	1.00|1.00	0.50	0.60	**0.69**|0.68
	AUPR	1.00	1.00	1.00|1.00	0.89	0.93	**0.95**|0.94
2	AUC	0.96	0.97	**0.99**|**0.99**	**0.88**	0.83	0.75|0.79
	AUPR	0.99	**1.00**	**1.00**|**1.00**	**0.98**	0.97	0.95|0.96
3	AUC	1.00	1.00	1.00|1.00	0.61	**0.68**	0.63|0.63
	AUPR	1.00	1.00	1.00|1.00	0.93	**0.94**	0.93|0.93
4	AUC	0.98	0.98	**1.00**|**1.00**	**0.98**	0.97	0.93|0.94
	AUPR	1.00	1.00	1.00|1.00	**1.00**	**1.00**	0.99|0.99
5	AUC	1.00	1.00	1.00|1.00	0.68	**0.80**	0.75|0.75
	AUPR	1.00	1.00	1.00|1.00	0.95	**0.97**	0.96|0.96
6	AUC	0.88	0.90	0.95|**0.96**	0.89	**0.93**	0.91|0.92
	AUPR	0.98	0.99	0.99|**1.00**	0.99	0.99	0.99|0.99
7	AUC	1.00	1.00	1.00|1.00	0.58	0.64	**0.66**|0.65
	AUPR	1.00	1.00	1.00|1.00	0.93	**0.94**	**0.94**|**0.94**
8	AUC	0.74	0.86	**0.99**|**0.99**	**0.99**	0.97	0.96|0.97
	AUPR	0.94	0.98	**1.00**|**1.00**	1.00	1.00	1.00|1.00
9	AUC	1.00	1.00	1.00|1.00	0.51	0.81	**0.82**|0.81
	AUPR	1.00	1.00	1.00|1.00	0.90	**0.98**	**0.98**|**0.98**

Table A.8: OOD detection on EEG and ECG using RealNVP. AUROC and AUPR are shown (higher is better). The batch size is fixed as 10. Our results are denoted by GOOD1KS | GOOD2KS.

InD/OOD	TyTest		KLOD		Ours	
	AUC	AUPR	AUC	AUPR	AUC	AUPR
EEG/ECG	0.15	0.33	0.60	0.50	0.50|0.92	0.47|0.88
ECG/EEG	1.00	1.00	0.99	0.99	0.90|0.92	0.92|0.94

Appendix B

Appendix of Chapter 3

B.1 Proof

Theorem 5. *Let $\mathcal{M}_i$ $(i = 1, 2, \ldots, k)$ be k DP mechanisms, and each $\mathcal{M}_i$ satisfies (ϵ_i, δ_i)-DP. Given a deterministic partitioning function f, let $D_1, D_2, \ldots, D_k$ be the disjoint partitions by executing f on D. Releasing $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_k(D_k)$ satisfies $(\max_{i\in I} \epsilon_i, \max_{i\in I} \delta_i)$-DP, where $I = \{1, 2, \ldots, k\}$.*

Proof. Without loss of generality, given two neighboring datasets D and D', assume that D contains one more element than D'. Executing f on D and D', we have partitions $D_1, D_2, \ldots, D_k$ and $D'_1, D'_2, \ldots, D'_k$, respectively. There exists j such that (1) D_j contains one more element than D'_j, and (2) $D_s = D'_s$ for $s = 1, 2, \ldots, k$ and $s \neq j$. Denote $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_k(D_k)$ by $\mathcal{M}(D)$. Since the subsets are disjoint from each other, running k algorithms on each subset is independent of each other. For any sequence $t = (t_1, t_2, \ldots, t_k)$ of outputs of $\mathcal{M}_1, \ldots, \mathcal{M}_k$ where $t_i \in \mathit{Range}(\mathcal{M}_i)$, we have:

$$
\begin{aligned}
\Pr[\mathcal{M}(D) = t] &= \Pr[\mathcal{M}_1(D_1) = t_1 \wedge \mathcal{M}_2(D_2) = t_2 \wedge \ldots \wedge \mathcal{M}_k(D_k) = t_k] \\
&= \Pr[\mathcal{M}_j(D_j) = t_j] \prod_{s=1,2,\ldots,k,s\neq j} \Pr[\mathcal{M}_s(D_s) = t_s] \\
&\leq \big(\exp(\epsilon_j)\Pr[\mathcal{M}_j(D'_j) = t_j] + \delta_j\big) \prod_{s=1,2,\ldots,k,s\neq j} \Pr[\mathcal{M}_s(D'_s) = t_s] \\
&= \exp(\epsilon_j) \prod_{i=1,2,\ldots,k} \Pr[\mathcal{M}_i(D'_i) = t_i] + \delta_j \prod_{s=1,2,\ldots,k,s\neq j} \Pr[\mathcal{M}_s(D'_s) = t_s] \\
&= \exp(\epsilon_j)\Pr[\mathcal{M}(D') = t] + \delta_j \prod_{s=1,2,\ldots,k,s\neq j} \Pr[\mathcal{M}_s(D'_s) = t_s] \\
&\leq \exp(\epsilon_j)\Pr[\mathcal{M}(D') = t] + \delta_j \\
&\leq \exp(\max_{i=1,2,\ldots,k} \epsilon_i)\Pr[\mathcal{M}(D') = t] + \max_{i=1,2,\ldots,k} \delta_i
\end{aligned}
$$

□

B.2 Datasets

We briefly introduce the public datasets and associated preprocessing. Image size is shown in $\#channels \times height \times width$. Images are normalized to the range of $[0, 1]$.

MNIST (LeCun et al. 1998) & Fashion MNIST (Xiao et al. 2017): MNIST contains hand-written digits images, whereas Fashion MNIST contains cloth and shoe images. Images in both datasets are single-channel, in the size of $1 \times 28 \times 28$, and have 10 classes. We adopt the official training and test split. 10k images from the training split are randomly held out as the validation set.

CelebA (Liu, P. Luo, et al. 2015): CelebA is a dataset including face images of celebrities. Each image is in the size of $3 \times 178 \times 218$ and has 40 binary attributes. All images are cropped to $3 \times 178 \times 178$, and then resized to $3 \times 32 \times 32$. We also adopt the official training, validation and test split, but randomly select 50k images of each gender from the training split as our training set.

CelebA-HQ (Karras et al. 2018): CelebA-HQ is a high-quality version of CelebA, which is commonly recognized as a high-resolution image set. It consists of 30k images in total. We download a gender conditioned split (where images are resized to $3 \times 256 \times 256$) by following this link. 1999 images are randomly held out from the training split as the validation set.

B.3 Rényi differential privacy (RDP)

Rényi differential privacy (RDP) extends ordinary DP using Rényi's α divergence (Rényi 1961) and provides tighter and easier composition property than the ordinary DP notion, thus we adopt RDP to accumulate the privacy cost. Importantly, RDP is linearly composable, and a mechanism satisfying (α, ϵ)-RDP also satisfies$(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta)$-DP for any $\delta \in (0, 1)$.

DP-SGD tracks the total privacy consumption as follows: (1) for each training iteration, compute the RDP privacy cost for a subsampled batch where Gaussian mechanism is applied; (2) compose RDP mechanisms over training iterations; (3) convert RDP back to (ϵ, δ)-DP. The implementation details are given below.

B.4 Framework

The schematic of DP-LFlow is shown in Figure B.1.

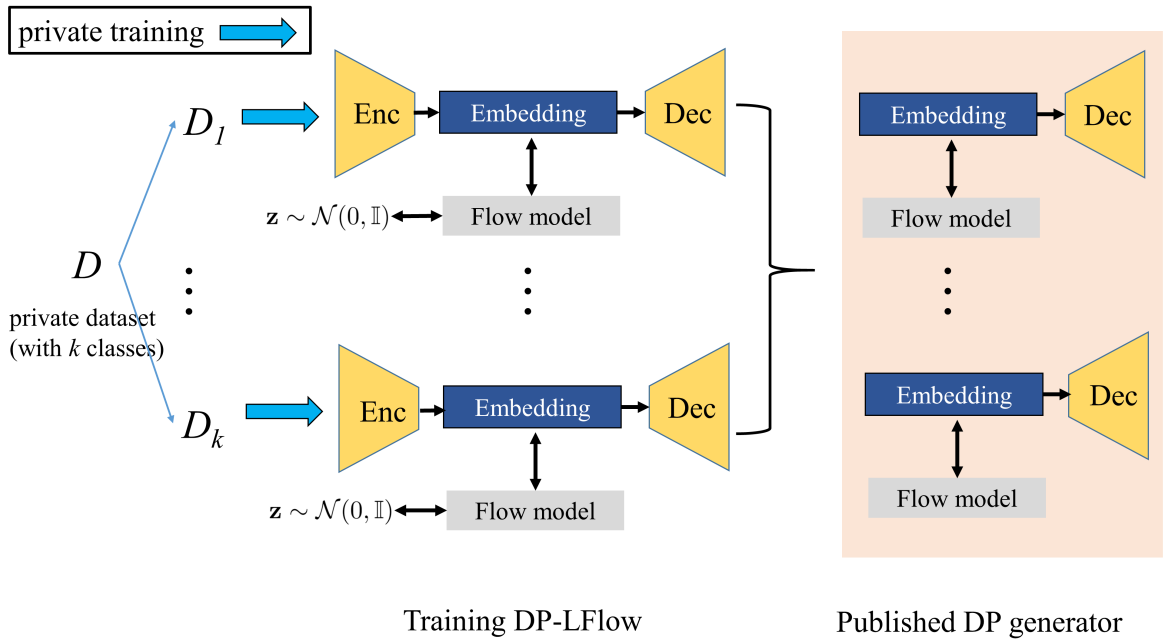


Figure B.1: The framework of DP-LFlow

B.5 Implementation

B.5.1 Flow models

Our code for flow generative models are adapted from public repos, i.e. Glow and RealNVP. Hyperparameters of the network are selected by comparing the performance on the validation set, and the selection results are given in Table B.1.

Table B.1: Network configurations for different datasets in the experiments. $\#h\_conv$ denotes the number of hidden sizes in the convolutional layers. $\#h\_lin$ denotes the number of hidden sizes in the linear layers of flow. $\#c$ denotes the length of latent code. $\#b$ means the number of blocks in flow.

Dataset	$\#h\_conv$ in encoder	$\#h\_conv$ of decoder	$\#c$	$\#b$	$\#h\_lin$
MNIST	[32, 64]	[64, 32]	20	9	200
FMNIST	[32, 64]	[64, 32]	20	9	200
CelebA	[64, 128, 256]	[256, 128, 64]	32	9	200
CelebA-HQ	[16, 32, 64, 128, 256, 512]	[512, 256, 128, 64, 32, 16]	64	12	256

B.5.2 Privacy implementation

We use a public repo, i.e. pyvacy, for implementing DP-SGD algorithm, as well as the total privacy calculation. Pyvacy tracks the privacy loss by RDP accountant, which is a PyTorch implementation based on Tensorflow Privacy.

For all datasets we use, we set subsampling rate as 0.1, training iterations as 300, noise multiplier as 1.25 to target $(10, 10^{-5})$-DP and 4.5 to target $(1, 10^{-5})$-DP, respectively. With better evaluation performance on the validation set, gradient clipping norms are set as 0.1 for MNIST and Fashion MNIST, 0.01 for CelebA, and 10 for CelebA-HQ.

B.5.3 Fréchet Inception Distance (FID)

FID calculates the distance between the feature vectors extracted by InceptionV3 pool3 layer (Szegedy et al. 2016) on real and synthetic samples. Specifically,

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$$

where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are activations of InceptionV3 pool3 layer of real images and generated images, respectively, and $\text{Tr}(A)$ refers to the trace of a matrix A. Intuitively, a lower FID means the generation X_g is more realistic (or more similar to X_r). We use a PyTorch implementation for computing FID, which will resize images and repeat channels three times for grayscale images to meet the input size requirement.

B.5.4 Classification task

We follow T. Cao et al. (2021) for the classifier implementation. We import scikit-learn package for implementation logistic regression classifier (e.g. from sklearn.linear_model import LogisticRegression) with default parameter settings.

The MLP network consists of following layers: linear($input\_dim, 100$) $\rightarrow$ ReLU $\rightarrow$ linear(100, $output\_dim$) $\rightarrow$ Softmax.

The CNN consists of following layers: Conv2d($input\_channels$, 32, kernel_size=3, stride = 2, padding=1) $\rightarrow$ Dropout(p=0.5) $\rightarrow$ ReLU $\rightarrow$ Conv2d(32, 64, kernel_size=3, stride = 2, padding=1) $\rightarrow$ Dropout(p=0.5) $\rightarrow$ ReLU $\rightarrow$ flatten $\rightarrow$ linear(in_dim, out_dim) $\rightarrow$ Softmax.

Both MLP and CNN are optimized by Adam with default parameters. All classifiers are trained on synthetic data, and we report test accuracy on real test data as the evaluation metric.

B.5.5 Baselines

All results of DP-MERF (Harder et al. 2021) are obtained by running their code with default parameters. It is worth mentioning that DP-MERF does not implement on CelebA. We adapt their code on CelebA by using the generative network they designed for SVHN with $16, 8, 8$ channels for three convolutional layers, respectively.

GS-WGAN only implements on $(10, 10^{-5})$-DP. To target $(1, 10^{-5})$-DP, we tried two vanilla variations by tuning parameters of $(10, 10^{-5})$-DP in their code, i.e. increasing noise scale while keeping the rest parameters unchanged, or decreasing the number of iterations while keeping the rest parameters unchanged. We experimentally found that both variations will not generate meaningful images. The former variation is not even able to generate anything on Fashion MNIST, so we instead present the latter variation for comparison.

All other results (e.g. numbers in the tables, generated images) are cited from papers as we specify.

Appendix C

Appendix of Chapter 4

C.1 Proof

Theorem 7 (Parallel composition of v-RDP). *If mechanism $\mathcal{M}_i$ satisfies (α, ϵ_i)-RDP for $i = 1, 2, \ldots, m$, and let $D_1, D_2, \ldots, D_m$ be the disjoint partitions by executing a deterministic partitioning function P on D. Releasing $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_m(D_m)$ satisfies $(\alpha, \max_{i \in I} \epsilon_i)$-RDP, where $I = \{1, 2, \ldots, m\}$.*

Proof. Without loss of generality, given two neighboring datasets D and D', assume that D contains one more element than D'. Executing P on D and D', we have partitions $D_1, D_2, \ldots, D_m$ and $D'_1, D'_2, \ldots, D'_m$, respectively. There exists j such that (1) D_j contains one more element than D'_j, and (2) $D_s = D'_s$ for $s = 1, 2, \ldots, m$ and $s \neq j$. Denote $\mathcal{M}_1(D_1), \ldots, \mathcal{M}_m(D_m)$ by $\mathcal{M}(D)$. Using additivity of Rényi divergence in (Van Erven and Harremos 2014) (Thm 28):

$$\begin{aligned}
\mathbb{D}_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) &= \sum_{i=1}^{m} \mathbb{D}_\alpha(\mathcal{M}_i(D_i)||\mathcal{M}_i(D'_i)) \\
&= \sum_{i=1,\ldots,m, i \neq j} \mathbb{D}_\alpha(\mathcal{M}_i(D_i)||\mathcal{M}_i(D'_i)) + \mathbb{D}_\alpha(\mathcal{M}_j(D_j)||\mathcal{M}_j(D'_j)) \\
&\leq \epsilon_j \leq \max_{i=1,2,\ldots,m} \epsilon_i.
\end{aligned}$$

The proof is complete. □

Theorem 8 (Post-processing theorem of f-RDP). *If a function f_D is (α, ϵ)-RDP, so is $g \circ f_D$, where g is a post-processing mechanism that only depends on a finite number of outputs of f_D.*

Proof. Given any finite subsets S, we reach the reduction of proof of the post-processing theorem in (Mironov 2017):

$$\mathbb{D}_\alpha(f_D(S)||f_{D'}(S)) \geq \mathbb{D}(g(f_D(S))||g(f_{D'}(S))).$$

□

Lemma 1. *Definition 7 implies:*

$$\mathbb{P}(\widetilde{f_D} \in A) \leq \left(\exp(\epsilon)\mathbb{P}(\widetilde{f_{D'}} \in A)\right)^{\frac{\alpha-1}{\alpha}}, \quad \forall A \in \mathcal{F}_0, \tag{C.1}$$

where we reuse all the notations in Definition 6.

Proof. Proof sketch: we will show that Eq. equation C.1 holds whenever Eq. equation 4.4 holds.

By taking logarithm and rearranging from Eq. equation C.1, we have

$$\frac{1}{\alpha-1}\log\left(\left(\frac{\mathbb{P}(\widetilde{f_D} \in A)}{\mathbb{P}(\widetilde{f_{D'}} \in A)}\right)^{\alpha}\mathbb{P}(\widetilde{f_{D'}} \in A)\right) \leq \epsilon. \tag{C.2}$$

By the definition of cylinder sets, $\widetilde{f_D} \in A$ implies that $\widetilde{f_D}(S)$ is in some sets B for any finite subsets $S = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of T. Thus,

$$\mathbb{P}(\widetilde{f_D} \in A) = \mathbb{P}(\widetilde{f_D}(S) \in B) = \int_S p(x)\,\mathrm{d}\mu(x).$$

where p is the density of $\widetilde{f_D}(S)$. Now we can translate Eq. equation C.2 into

$$\frac{1}{\alpha-1}\log\left(\left(\frac{\int_S p(x)\,\mathrm{d}\mu(x)}{\int_S q(x)\,\mathrm{d}\mu(x)}\right)^{\alpha}\int_S q(x)\,\mathrm{d}\mu(x)\right) \leq \epsilon. \tag{C.3}$$

Recall that Eq. equation 4.4 in Definition 7 can be translated to:

$$\frac{1}{\alpha-1}\log\left(\int\left(\frac{p(x)}{q(x)}\right)^{\alpha}q(x)\,\mathrm{d}\mu(x)\right) \leq \epsilon.$$

Note that p, q are non-negative, so $\int(\frac{p(x)}{q(x)})^{\alpha}q(x)\,\mathrm{d}\mu(x) \geq \int_S(\frac{p(x)}{q(x)})^{\alpha}q(x)\,\mathrm{d}\mu(x)$. Compared to Eq. equation C.3, it now suffices to show

$$\int_S\left(\frac{p(x)}{q(x)}\right)^{\alpha}q(x)\,\mathrm{d}\mu(x) \geq \left(\frac{\int_S p(x)\,\mathrm{d}\mu(x)}{\int_S q(x)\,\mathrm{d}\mu(x)}\right)^{\alpha}\int_S q(x)\,\mathrm{d}\mu(x).$$

Define $p^*(x) = \frac{p(x)\mathbb{1}(x\in S)}{\int_S p(x)\,\mathrm{d}\mu(x)}$, and q^* is similarly defined. All we want to show reduces to

$$\int\left(\frac{p^*(x)}{q^*(x)}\right)^{\alpha}q^*(x)\,\mathrm{d}\mu(x) \geq 1 \iff \mathbb{E}_{x\sim q^*}[(\frac{p^*}{q^*})^{\alpha}] \geq \left(\mathbb{E}_{x\sim q^*}[\frac{p^*}{q^*}]\right)^{\alpha} = 1,$$

where the final step follows from Jensen's inequality. □

Proposition 3 (f-RDP conversion to f-DP)**.** *A function $\widetilde{f_D}$ that is (α, ϵ)-RDP is $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-DP.*

Proof. We reuse the notations in Definition 6 in this proof. To show that an (α, ϵ)-RDP function satisfies (ϵ', δ)-DP for functions, where $\epsilon' = \epsilon + \frac{\log 1/\delta}{\alpha - 1}$, the objective becomes to show $\mathbb{P}[\widetilde{f_D} \in A] \leq \exp(\epsilon') \times \mathbb{P}[\widetilde{f_{D'}} \in A] + \delta$. With the help of Lemma 1, denote $\mathbb{P}[\widetilde{f_{D'}} \in A]$ by Q and we perform a case analysis:

- If $\exp(\epsilon)Q \leq \delta^{\alpha/(\alpha-1)}$, then

$$\mathbb{P}(\widetilde{f_D} \in A) \leq \big(\exp(\epsilon)Q\big)^{\frac{\alpha-1}{\alpha}} \leq \delta \leq \exp(\epsilon')Q + \delta.$$

- If $\exp(\epsilon)Q > \delta^{\alpha/(\alpha-1)}$, then

$$\begin{aligned}
\mathbb{P}(\widetilde{f_D} \in A) &\leq \big(\exp(\epsilon)Q\big)^{\frac{\alpha-1}{\alpha}} \\
&= \exp(\epsilon)Q\big(\exp(\epsilon)Q\big)^{-\frac{1}{\alpha}} \\
&\leq \exp(\epsilon)Q \cdot \delta^{-\frac{1}{\alpha-1}} \\
&= \exp(\epsilon + \frac{\log 1/\delta}{\alpha - 1})Q \\
&\leq \exp(\epsilon')Q + \delta.
\end{aligned}$$

Combining the two cases completes the proof. □

Theorem 9 (Parallel composition of f-RDP)**.** *Given a deterministic partitioning function P, let $D_1, D_2, \ldots, D_m$ be the disjoint partitions by executing P on D. If function f_{D_i} satisfies (α, ϵ_i)-RDP for $i = 1, 2, \ldots, m$, releasing $(f_{D_1}, \ldots, f_{D_m}) := f_D$ satisfies $(\alpha, \max_{i \in I} \epsilon_i)$-RDP, where $I = \{1, 2, \ldots, m\}$.*

Proof. Without loss of generality, given two neighboring datasets D and D', assume that D contains one more element than D'. Executing P on D and D', we have partitions $D_1, D_2, \ldots, D_m$ and $D'_1, D'_2, \ldots, D'_m$, respectively. There exists j such that (1) D_j contains one more element than D'_j, and (2) $D_s = D'_s$ for $s = 1, 2, \ldots, m$ and $s \neq j$. Given any finite subsets $S = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \subset T$, Using additivity of Rényi divergence in (Van Erven and Harremos 2014) (Thm 28), we have:

$$\begin{aligned}
\mathbb{D}_\alpha(f_D(S)||f_{D'}(S)) &= \sum_{i=1}^{m} \mathbb{D}_\alpha(f_{D_i}(S)||f_{D'_i}(S)) \\
&= \sum_{i=1,\ldots,m, i \neq j} \mathbb{D}_\alpha(f_{D_i}(S)||f_{D'_i}(S)) + \mathbb{D}_\alpha(f_{D_j}(S)||f_{D'_j}(S)) \\
&\leq \epsilon_j \leq \max_{i \in \{1,2,\ldots,m\}} \epsilon_i.
\end{aligned}$$

□

Theorem 10 (Sequential composition of f-RDP). *Let $\{f_D : D \in \mathcal{D}\}$ and $\{g_D : D \in \mathcal{D}\}$ be two families of functions indexed by dataset D, where $f_D \in \mathcal{R}_1^T$ is (α, ϵ_1)-RDP and $g_D : \mathcal{R}_1^T \to \mathcal{R}_2^S$ is (α, ϵ_2)-RDP. Releasing the sequentially composed functional mechanism $h_D = (f_D, g_D \circ f_D) \in \mathcal{R}_1^T \times \mathcal{R}_2^S = (\mathcal{R}_1 \times \mathcal{R}_2)^{T \times S}$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Proof. According to Definition 7, our objective is to show for any finite subsets $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of T and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ of S,

$$\begin{aligned} & \mathbb{D}_\alpha(h_D(X,Y) || h_{D'}(X,Y)) \le \epsilon_1 + \epsilon_2 \\ \iff & \mathbb{D}_\alpha\Big(\big((f_D(X), g_D(f_D, Y)\big) || \big(f_{D'}(X), g_{D'}(f_{D'}, Y)\big)\Big) \le \epsilon_1 + \epsilon_2. \end{aligned}$$

Here we adapt the proof of Proposition 1 in (Mironov 2017). Let F be the distribution of $f_D(X)$, G be the distribution of $g_D(f_D, Y)$, $H = (F, G)$, and F', G', H' are similarly defined on adjacent dataset D'.

$$\begin{aligned} \exp[(\alpha-1)\mathbb{D}_\alpha(h_D(X,Y)||h_{D'}(X,Y))] &= \int_{\mathcal{R}_1 \times \mathcal{R}_2} H(x,y)^\alpha H'(x,y)^{1-\alpha} \,\mathrm{d}x \,\mathrm{d}y \\ &= \int_{\mathcal{R}_1} \int_{\mathcal{R}_2} [F(x)G(x,y)]^\alpha [F'(x)G'(x,y)]^{1-\alpha} \,\mathrm{d}x \,\mathrm{d}y \\ &= \int_{\mathcal{R}_1} F(x)^\alpha F'(x)^{1-\alpha} \Big[\int_{\mathcal{R}_2} G(x,y)^\alpha G'(x,y)^{1-\alpha} \,\mathrm{d}y \Big] \,\mathrm{d}x \\ &\le \int_{\mathcal{R}_1} F(x)^\alpha F'(x)^{1-\alpha} \cdot \exp[(\alpha-1)\epsilon_2] \\ &\le \exp[(\alpha-1)(\epsilon_1+\epsilon_2)]. \end{aligned}$$

□

Proposition 4 (Gaussian mechanism for f-RDP). *Let G be a sample path of a Gaussian process having mean zero and covariance function k. Let M denote the Gram matrix (as defined in Eq. 4.2). Let $\{f_D : D \in \mathcal{D}\}$ be a family of functions indexed by database D. Releasing $\widetilde{f_D} = f_D + \sigma\Delta \cdot G$ satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP whenever Eq. 4.3 holds.*

Proof. Consider any finite set $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in T^n$, the vector $(G(\mathbf{x}_1), \ldots, G(\mathbf{x}_n))$ follows a multivariate Gaussian with mean zero and covariance given by Eq. equation 4.2. Thus, evaluating $\widetilde{f_D}$ at any finite sets would form a vector that satisfies Eq. equation 4.5 in Definition 8, which completes the proof.

□

C.2 Datasets

MNIST (LeCun et al. 1998) & Fashion MNIST (Xiao et al. 2017): MNIST contains hand-written digits images, whereas Fashion MNIST contains cloth and shoe

images. Images in both datasets are single-channel, in the size of $1 \times 28 \times 28$, which are resized to $1 \times 32 \times 32$ and normalized to have 0.5 mean and 0.5 standard deviation. Both datasets have 10 classes. We adopt the official training and test split. MNIST and Fashion MNIST are made available under Creative Commons Attribution-Share Alike 3.0 license and MIT License, respectively.

CelebA (Liu, P. Luo, et al. 2015): CelebA is a dataset including face images of celebrities. Each image is in the size of $3 \times 178 \times 218$ and has 40 binary attributes. All images are center-cropped to $3 \times 178 \times 178$, then resized to $3 \times 32 \times 32$, and normalized to have 0.5 mean and 0.5 standard deviation. We also adopt the official training, validation and test split, but randomly select 60k images from the training split as our training set. The CelebA dataset is available for non-commercial research purposes only, as described on their website.

C.3 Implementation

Generative network: Our unconditional generative network is based on the official *code* of MMD-GAN (C.-L. Li et al. 2017). For the NN architecture, we use the same Conv2dTranpose layer with similar depth as prior related works. Besides, we use the same architecture for both gray-scale and RGB image sets to demonstrate its versatility (of course with different NN parameters to account for the difference in input channels). Encoding labels to the latent code leads to the conditional variant. All networks are optimized by RMSprop with a learning rate 5×10^{-5}.

CNN classifier: We follow T. Cao et al. (2021) for the classifier implementation, as described in Appendix B.5.4.

Privacy: We use *Tensorflow privacy* for computing the total privacy cost, which only requires inputting a few important parameters, e.g. subsampling rate (or batch size), noise multiplier, training epochs (iterations), target δ ($\delta = 10^{-5}$ in all our experiments). We summarize the parameters in Table C.1.

Baselines: We use the official code of DP-MERF to replicate their results under different ϵ and use the same evaluation metrics to add quantitative comparison. The numerical evaluations of rest baselines are cited from related works, as specified in the caption of the tables.

Table C.1: Training parameters for retaining DP on MNIST and Fashion MNIST. Both variants on CelebA use parameters in the row of Conditional. q is the subsampling rate, and σ is the noise multiplier.

	target ϵ	q	σ	epochs (iterations)
Conditional	10.0	0.001	0.60	200 (200k)
	1.0	0.001	1.95	200 (200k)
	0.2	0.001	8.00	200 (200k)
Parallel	10.0	0.010	1.00	200 (20k)
	1.0	0.010	5.75	200 (20k)
	0.2	0.010	25.00	200 (20k)

Computation resources: All computation is conducted by one NVIDIA T4 GPU. It costs 1.5 hours to train a conditional generator on MNIST (and Fashion MNIST) and around 5 hours on CelebA. Each sub-model in the parallel variant takes only 10 minutes to train on MNIST (and Fashion MNIST) and 30 minutes on CelebA. A better GPU should compute faster. As a reference, DP-Sinkhorn and GS-WGAN can take up to 24 hours to train, and GS-WGAN even requires at least 2 GPUs (as they have 1000 discriminators).

C.4 Additional Results

For completeness, we include the qualitative comparison between our method and related works under $(10, 10^{-5})$-DP on all three datasets in Figures C.1 and C.2. Numerical comparison is in Table C.2. We note that $\epsilon = 10$ is usually considered a weak privacy regime. Generally, all baselines can generate decent images when $\epsilon = 10$, whereas our method still generates more diverse and more informative images, especially on CelebA.

C.5 Broader Impact and Limitations

Broader impact: Some prior works reveal that DP training could introduce additional biases to the model, leading to fairness concerns (Bagdasaryan et al. 2019; Esipova et al. 2023; Xu, Du, et al. 2021). We did not explore this point in the current experiments as there are no sensitive attributes involved, but we can imagine the potential negative effects on fairness issues if our methods are not used properly.

Besides, we do not expect this work to trigger any differential privacy concern by its design, and we do not expect it to be used for Deepfakes (as the non-private counterparts are better in utility), but there might be some unintended uses that we are not aware of for now.

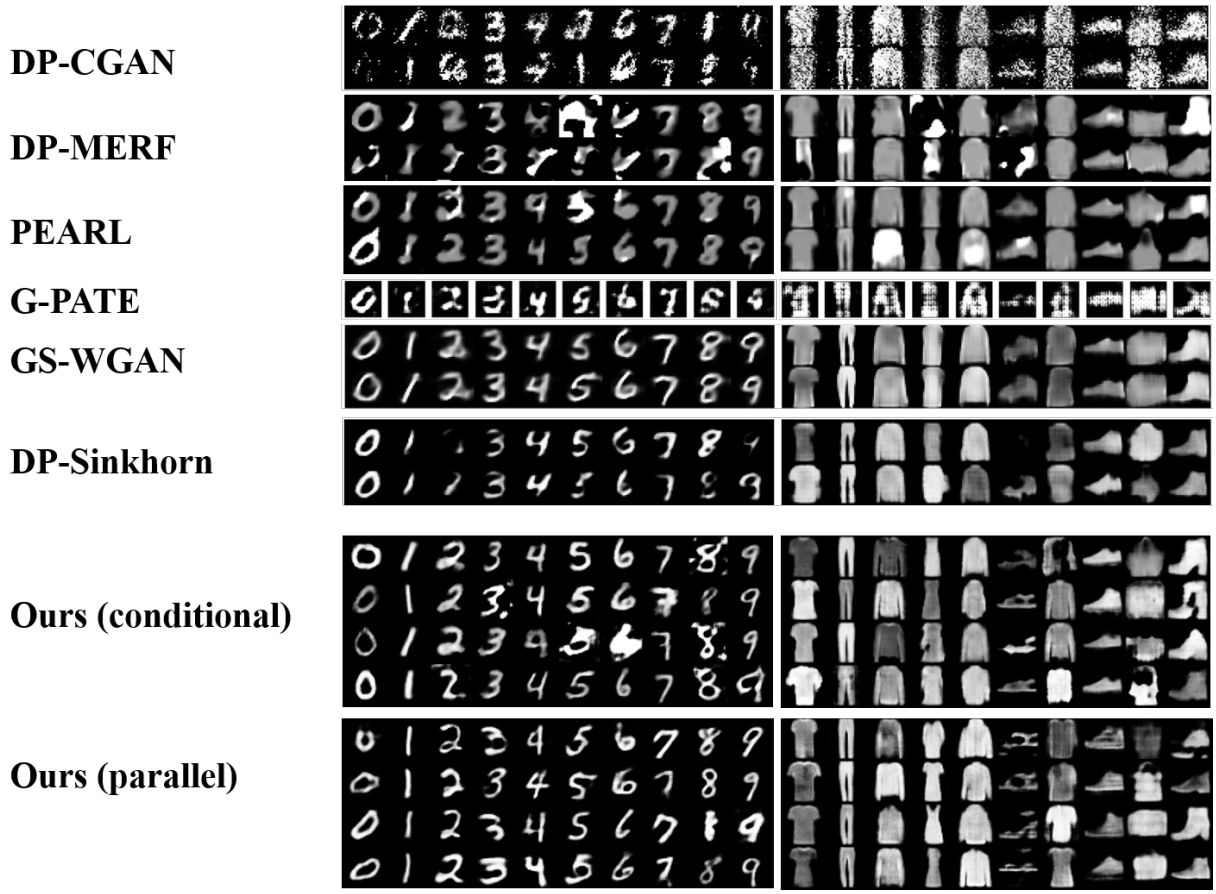


Figure C.1: Qualitative comparison under $(10, 10^{-5})$-DP on MNIST and Fashion MNIST



Figure C.2: Qualitative comparison under $(10, 10^{-5})$-DP on CelebA

Table C.2: Quantitative comparison under $(10, 10^{-5})$. Acc denotes classification accuracy, which is shown in %. ↑ and ↓ refer to higher is better or lower is better, respectively. Results of DP-CGAN, GS-WGAN, DP-Sinkhorn are cited from T. Cao et al. (2021) and Long et al. (2021). Results of PEARL and DPDM are cited from Dockhorn et al. (2023). Results of G-PATE and DataLens are cited from their papers, respectively. (*): DPDM did unconditional generation on CelebA.

Method	ϵ	MNIST		FMNIST		CelebA	
		FID ↓	Acc ↑	FID ↓	Acc ↑	FID ↓	Acc ↑
DPDM	10	5.0	97.3	18.6	84.9	*21.1	-
DP-CGAN	10	179.2	63.0	243.8	46.0	-	-
DP-MERF	10	121.4	82.0	110.4	73.2	211.1	64.0
PEARL	10	116.0	78.8	102.0	64.9	-	-
G-PATE	10	150.6	80.9	171.9	69.3	305.9	70.7
DataLens	10	173.5	80.7	167.7	70.6	320.8	72.9
GS-WGAN	10	61.3	80.0	131.3	65.0	432.5	63.3
DP-Sinkhorn	10	55.6	79.1	129.4	68.9	-	-
Ours (conditional)	10	29.9	93.1	56.3	79.2	77.2	85.8
Ours (parallel)	10	17.7	96.4	38.1	82.0	78.5	87.4

Limitations: The current conditional approach assumes a uniform distribution over different labels, which happens to hold for all three datasets. A more tailored algorithm is needed if the label distribution is imbalanced. The current parallel approach is not subject to a specific label distribution, but it may hinder its practical deployment if the number of classes is too large.

C.6 A Simplified View of RDP for Vectors and Functions

In this section, we propose to define Rényi differential privacy (RDP) for vectors and functions in a unified way.

Let $\mathcal{D}$ be a collection of datasets and $f : \mathcal{D} \to \mathcal{R}$ be a *randomized* mechanism. We consider $\mathcal{R} = \mathbb{R}^{\mathcal{T}}$ so we may also identify f as a bivariate (random) function $f : \mathcal{T} \times \mathcal{D} \to \mathbb{R}$. In the following we call two datasets $D, D' \in \mathcal{D}$ adjacent if they differ by one data point. We recall the Rényi α-divergence (Rényi 1961) between two random variables X_1 and X_2 with density p and q, respectively:

$$\mathbb{D}_\alpha(\mathsf{X}_1 \| \mathsf{X}_2) := \mathbb{D}_\alpha\big(p(\mathbf{x}) \| q(\mathbf{x})\big) := \tfrac{1}{\alpha-1} \log \mathbb{E}_{\mathsf{X}\sim q} \left[\tfrac{p(\mathsf{X})}{q(\mathsf{X})}\right]^\alpha \geq 0,$$

where $\alpha > 1$ is a hyperparameter and the last inequality follows from Jensen's inequality. Since we can rewrite

$$\mathbb{D}_\alpha(\mathsf{X}_1\|\mathsf{X}_2) = \tfrac{1}{\alpha-1}\log \mathbb{E}_{\mathsf{X}\sim p}\left[\tfrac{p(\mathsf{X})}{q(\mathsf{X})}\right]^{\alpha-1},$$

it is clear Rényi's α-divergence $\mathbb{D}_\alpha$ is increasing w.r.t. α, with the following limits:

$$\mathsf{KL}(\mathsf{X}_1\|\mathsf{X}_2) = \lim_{\alpha\to 1}\mathbb{D}_\alpha(\mathsf{X}_1\|\mathsf{X}_2) \le \lim_{\alpha\to\infty}\mathbb{D}_\alpha(\mathsf{X}_1\|\mathsf{X}_2) = \operatorname*{ess\,sup}_{\mathbf{x}} \log \tfrac{p(\mathbf{x})}{q(\mathbf{x})}.$$

For joint distributions, we can easily verify the following decomposition rule:

$$\begin{aligned}\mathbb{D}_\alpha(p(\mathbf{x},\mathbf{z})\|q(\mathbf{x},\mathbf{z})) &= \tfrac{1}{\alpha-1}\log \mathbb{E}_{\mathsf{X}\sim q}\left[\left[\tfrac{p(\mathsf{X})}{q(\mathsf{X})}\right]^\alpha \exp\Big((\alpha-1)\mathbb{D}_\alpha\big[p(\mathbf{z}|\mathsf{X})\|q(\mathbf{z}|\mathsf{X})\big]\Big)\right] \\ &\ge \mathbb{D}_\alpha\big(p(\mathbf{x})\|q(\mathbf{x})\big),\end{aligned} \tag{C.4}$$

where the inequality follows from the nonnegativity of the conditional Rényi divergence inside the exponential. In other words, the α-divergence between joint distributions is always larger than that between marginal distributions.

We are now ready to define Rényi differential privacy (Mironov 2017):

Definition 9 ((α,ϵ)-RDP)**.** *A randomized mechanism $f : \mathcal{T}\times\mathcal{D}\to\mathbb{R}$ is (α,ϵ)-RDP if for all adjacent datasets $D, D'\in\mathcal{D}$,*

$$\sup_{n\in\mathbb{N}}\sup_{\mathbf{t}:=[t_1,\dots,t_n]\subseteq\mathcal{T}} \mathbb{D}_\alpha\big(f(\mathbf{t};D)\|f(\mathbf{t};D')\big) \le \epsilon, \tag{C.5}$$

where $f(\mathbf{t};D) := [f(t_1;D),\dots,f(t_n;D)]$ is the concatenation of n random variables.

The acute reader will notice some immediate difference between our definition and the one in the literature, e.g., Mironov (2017, Definition 4). Before explaining our motivation, let us first notice that when $\mathcal{T} = [d] := \{1,\dots,d\}$, i.e., the range of our mechanism f is $\mathcal{R} = \mathbb{R}^d$ (which is the setting of Mironov 2017 and most existing work), the supremum in equation C.5 is attained when $n = d$ and $\mathbf{t} = [1,\dots,d]$, thanks to inequality equation C.4. In other words, the supremum in equation C.5 is superficial when $\mathcal{T}$ is a finite set. The important point is that we can now extend RDP effortlessly to any index set $\mathcal{T}$, including when $\mathcal{T}$ is even uncountable. For instance, in our later application to generative modeling (Section 4.5), we will use $\mathcal{T} = \mathbb{R}^m$. On the other hand, one might be tempted to lump all random variables together into $f(\mathcal{T};D) = \{f(t;D) : t\in\mathcal{T}\}$, hoping to obviate the supremum in equation C.5. However, this approach will run into two difficulties: Firstly, the rigorous minds will start to worry about measurability issues around $f(\mathcal{T};D)$ (as a stochastic process) and involve tedious measure-theoretic jargon (such as the cylinder field in Hall et al. 2013). Secondly, we no longer possess a dominating measure (e.g., Lebesgue) to define an easily computable density of the process $f(\mathcal{T};D)$. In contrast, our reduction to the finite case through taking the supremum does not suffer from either issue, and, as we will see, renders extensions of existing properties of RDP (to an arbitrary index set $\mathcal{T}$) rather natural or even trivial.

The following result was proved by Mironov (2017, Proposition 3):

Theorem 11. *If a mechanism f is (α, ϵ)-RDP, then for any (measurable) set A,*

$$\Pr[f(D) \in A] \leq \{\exp(\epsilon) \Pr[f(D') \in A]\}^{1-\frac{1}{\alpha}} \leq (1 - \tfrac{1}{\alpha}) \exp(\epsilon + \tfrac{1}{\alpha-1} \log \tfrac{1}{\delta}) \Pr[f(D') \in A] + \tfrac{\delta}{\alpha}.$$

In particular, f is $(\epsilon + \frac{1}{\alpha-1} \log \frac{1}{\delta}, \frac{\delta}{\alpha})$-DP.

Proof. For $\alpha \geq 1$, consider the (jointly) convex function (as the perspective of $p \mapsto p^\alpha$):

$$g(p, q) := q \left(\tfrac{p}{q}\right)^\alpha,$$

as well as the probability measure $\mathbf{1}_{\mathbf{x} \in A} \cdot \mu(\mathrm{d}\mathbf{x}) / \mu(A)$ (recall that $\mu(\mathrm{d}\mathbf{x})$ is the underlying dominating measure that the densities p and q are defined w.r.t.). Applying Jensen's inequality we obtain

$$\begin{aligned} \tfrac{1}{\mu(A)} \cdot \int_A q(\mathbf{x}) \left(\tfrac{p(\mathbf{x})}{q(\mathbf{x})}\right)^\alpha \mu(\mathrm{d}\mathbf{x}) &\geq g\left(\tfrac{1}{\mu(A)} \int_A p(\mathbf{x}) \mu(\mathrm{d}\mathbf{x}), \tfrac{1}{\mu(A)} \int_A q(\mathbf{x}) \mu(\mathrm{d}\mathbf{x})\right) \\ &= \tfrac{1}{\mu(A)} Q(A) \left(\tfrac{P(A)}{Q(A)}\right)^\alpha, \end{aligned}$$

where $P(A) := \int_A p(\mathbf{x}) \mu(\mathrm{d}\mathbf{x}) = \Pr[f(D) \in A]$ and similarly for $Q(A)$. Thus,

$$\epsilon \geq \mathbb{D}_\alpha(p \| q) \geq \tfrac{1}{\alpha-1} \log \int_A q(\mathbf{x}) \left(\tfrac{p(\mathbf{x})}{q(\mathbf{x})}\right)^\alpha \mu(\mathrm{d}\mathbf{x}) \geq \tfrac{1}{\alpha-1} \log \left[Q(A) \left(\tfrac{P(A)}{Q(A)}\right)^\alpha\right].$$

Rearranging we obtain the first inequality. The second inequality is simply the arithmetic-geometric mean inequality, after noting that

$$[\exp(\epsilon) Q(A)]^{1-\frac{1}{\alpha}} = \left[\exp(\epsilon + \tfrac{1}{\alpha-1} \log \tfrac{1}{\delta}) Q(A)\right]^{1-\frac{1}{\alpha}} \cdot \delta^{\frac{1}{\alpha}}.$$

Lastly, we relax the factor $(1 - \frac{1}{\alpha}) \leq 1$ to obtain the claimed DP guarantee. □

For two Gaussian distributions $\mathcal{N}(\mathbf{u}, \Sigma)$ and $\mathcal{N}(\mathbf{v}, \Sigma)$, their Rényi α-divergence can be readily computed from Liese and Vajda (1987, Proposition 2.22):

$$\mathbb{D}_\alpha\left(\mathcal{N}(\mathbf{u}, \Sigma) \| \mathcal{N}(\mathbf{v}, \Sigma)\right) = \tfrac{\alpha}{2} \left\langle \mathbf{u} - \mathbf{v}, \Sigma^{-1}(\mathbf{u} - \mathbf{v}) \right\rangle.$$

Therefore, the Gaussian mechanism

$$f(D) := F(D) + \sigma \cdot \mathcal{N}(\mathbf{0}, \Sigma)$$

is (α, ϵ)-RDP, where

$$\epsilon = \tfrac{\alpha}{2\sigma^2} \cdot \sup_{D \sim D'} \left\langle F(D) - F(D'), \Sigma^{-1}(F(D) - F(D')) \right\rangle.$$